

IEEE Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Equipment and Test Information via XML: Exchanging Test Descriptions

IEEE Standards Coordinating Committee 20

Sponsored by the
IEEE Standards Coordinating Committee 20 on
Test and Diagnosis for Electronic Systems

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 1671.1™-2009

1 February 2013

IEEE Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Equipment and Test Information via XML: Exchanging Test Descriptions

Sponsor

**IEEE Standards Coordinating Committee 20 on
Test and Diagnosis for Electronic Systems**

Approved 11 September 2009

IEEE-SA Standards Board

Approved as a Full-Use Standard 5 December 2012

IEEE-SA Standards Board

Abstract: This document specifies an exchange format, using the eXtensible Markup Language (XML), for exchanging the test description information defining test performance, test conditions, diagnostic requirements, and support equipment to locate, align, and verify the proper operation of a unit under test (UUT). This is in support of the development of test program sets (TPSs) that will be used in an automatic test environment.

Keywords: automatic test equipment (ATE), Automatic Test Markup Language (ATML), Automatic Test Markup Language (ATML) instance document, automatic test system (ATS), diagnostic requirements, IEEE 1671.1, test description, Test Program Set (TPS), test requirements, Test Requirements Document (TRD), XML schema

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2013 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1 February 2013. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

W3C is a registered trademark in the U.S. Patent & Trademark Office, owned by the World Wide Web Consortium

PDF: ISBN 978-0-7381-8152-3 STD98102
Print: ISBN 978-0-7381-8153-0 STDPD98102

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Notice and Disclaimer of Liability Concerning the Use of IEEE Documents: IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon any IEEE Standard document.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied "AS IS."

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

Translations: The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official Statements: A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on Standards: Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important to ensure that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. Any person who would like to participate in evaluating comments or revisions to an IEEE standard is welcome to join the relevant IEEE working group at <http://standards.ieee.org/develop/wg/>.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Photocopies: Authorization to photocopy portions of any individual standard for internal or personal use is granted by The Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

This introduction is not part of IEEE Std 1671.1-2009, IEEE Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Equipment and Test Information via XML: Exchanging Test Descriptions.

The benefits of using standards in test-related applications have long been recognized. The scope for standardization extends from low-level standards associated with test instrument control to high-level standards associated with specifying tests in an implementation-independent manner. In the 1960s, Aeronautical Radio, Incorporated started the development of the Abbreviated Test Language for Avionics Systems. In 1976, management of the ATLAS standard was passed to the IEEE, and the ATLAS acronym was changed to Abbreviated Test Language for All Systems to reflect its broader field of applications.

Within the IEEE, development of ATLAS and ATLAS-related standards was vested in an ad hoc committee, which later became the IEEE Standards Coordinating Committee 20 (SCC20). In the mid-1980s, SCC20 broadened the scope of its activities to embrace other standards projects related to test and diagnosis, including Automatic Test Program Generation (ATPG), Test Equipment Description Language (TEDL), Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE), A Broad Based Environment for Test (ABBET), Software Interface to Maintenance Information and Collection Analysis (SIMICA), Receiver Fixture Interface (RFI), Signal and Test Definition (STD), and Automatic Test Markup Language (ATML).

The parent standard, IEEE Std 1671^{TMab}, provides the framework for a family of standards providing specifications for test-related applications and environments. This family specifies language-independent elements within a wide variety of test environments, including built-in test systems, automatic test systems (ATS), and manual test environments. Each of these interfaces is specified in the form of an eXtensible Markup Language (XML) schema.

This child, or “dot,” standard, also known as an ATML component standard, specifies a new XML schema that provides for the representation of test description information.

XML schemas define the basic information required within any test application and provide a vehicle for formally defining the test environment by defining a class hierarchy corresponding to these basic information entities and providing several methods within each to enable basic operations to be performed on these entities. ATML component standards within the ATML framework define the particular requirements within the test environment.

Notice to users

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

^a IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (<http://standards.ieee.org/>).

^b Information on references can be found in Clause 2.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at <http://standards.ieee.org/index.html> or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit IEEE-SA Website at <http://standards.ieee.org/index.html>.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board, the Test and ATS Description (TAD) Subcommittee had the following membership:

Ashley Hulme, *Co-chair*

Ion Neag, *Co-chair*

Peter Abendroth
Jean-Yves Allard
Anthony Alwardt
Jack Amsell
Michael Araiza
Wesley Barnishan
Malcolm Brown
Gerald (Scott) Brown
Steve Cmiel
Bernard Dathy
Timothy Davis
David Droste
James Dumser
Gail Edwards
Tamara Einspanjer
William Eklow
Keith Ellis
Anthony Estrada
Anthony Faucogney
Jennifer Fetherman
William (Brit) Frank
Thomas Gauntner
Scott Gearhart
George Geathers
Anthony Geneva
William Gerstein
José Manuel Gonzalez
Chris Gorringer
Michael Granieri

Christophe Grard
Jay Guruswamy
Michelle Harris
Michael Harrison
David Heck
Alicia Helton
Hans Hopf
Orman (Bob) Horton
Li Hui
Ulla Jacobsson
Anand Jain
Carey Jimmerson
Mark Kaufman
Dexter Kennedy
John Knowles
Arthur Larsen
Lars-Olov Larsson
Adam Ley
Jay Lindsey
Teresa Lopes
William Maciejewski
Michael Malesich
Robert McGarvey
Alex Meloni
Scott Misha
Mukund Modi
Angela Nielson
Leslie Orlidge

Duy-Huan Pham
Dan Pleasant
Hugh Pritchett
David Ptacek
Peter Richardson
Rudy Ridwan
David Rohacek
Michael Rutledge
Paul Salopek
Howard Savage
Michel Schieber
Mike Seavey
Jamie Shand
David Sharone
John Sheppard
Mark Skiba
Roger J. Sowada
Robert Spinner
John Stabler
Joseph Stanco
Michael Stora
Walter Struppler
Ronald Taylor
Laurence Thayer
Henry Thomas
Vincent Tume
Joerg Urban
Timothy Wilmering
Pelín Yalcin

The following members of the balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Malcom Brown
Keith Chow
He Chun
Kevin Coggins
Tommy Cooper
David Droste
William (Brit) Frank
Scott Gearhart
George Geathers
William Gerstein
Sergiu Goma
José Manuel Gonzalez
Chris Gorringer
Randall Groves

Werner Hoelzl
Ashley Hulme
Anand Jain
Mark Kaufman
John Knowles
Adam Ley
Teresa Lopes
Robert McGarvey
Scott Misha
Prajit Nair
Ion Neag
Leslie Orlidge
Stephen Osella
Ulrich Pohl
John Ralph

Peter Richardson
Robert Robinson
David Rohacek
Bartien Sayogo
Mike Seavey
James E. Smith
Roger J. Sowada
Robert Spinner
Joseph Stanco
Walter Struppler
Ronald Taylor
Vincent Tume
Timothy Wilmering
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 11 September 2009, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

John Barr
Karen Bartleson
Victor Berman
Ted Burse
Richard DeBlasio
Andy Drozd
Mark Epstein

Alexander Gelman
Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick

David J. Law
Ted Olsen
Glenn Parsons
Ronald C. Petersen
Narayanan Ramachandran
Jon Walter Rosdahl
Sam Sciacca

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Howard L. Wolfman, *TAB Representative*
Michael Janezic, *NIST Representative*
Satish K. Aggarwal, *NRC Representative*

Lorraine Patsco
IEEE Standards Program Manager, Document Development

Bill Ash
IEEE Standards Program Manager, Technical Program Development

Contents

1. Overview	1
1.1 General	1
1.2 Scope	1
1.3 Purpose	2
1.4 Application	2
1.5 Conventions used within this document	2
2. Normative references.....	4
3. Definitions, acronyms, and abbreviations	4
3.1 Definitions	4
3.2 Acronyms and abbreviations	5
4. Schema—TestDescription.xsd	5
4.1 General	5
4.2 Elements	6
4.3 Simple types	12
4.4 Complex types	14
5. Conformance	147
6. Extensibility.....	148
Annex A (informative) TestDescription instance documents (.XML files)	149
A.1 TestDescription.....	149
Annex B (informative) Users information and examples	151
B.1 Example UUT	151
B.2 Example 1	153
B.3 Example 2	170
B.4 Example 3	177
B.5 Example 4	180
B.6 Other examples	181
Annex C (informative) Referenced IEEE standards.....	183
Annex D (informative) Glossary	184
Annex E (informative) Bibliography.....	185

IEEE Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Equipment and Test Information via XML: Exchanging Test Descriptions

IMPORTANT NOTICE: IEEE Standards documents are not intended to ensure safety, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

1.1 General

The family of Automatic Test Markup Language (ATML) standards is being developed under the guidance of the IEEE Standards Coordinating Committee 20 (SCC20) to serve as standards for product test. The ATML family of standards specifies a comprehensive environment for integrating design data, test strategies, test requirements, test procedures, test results management, and test system implementations. The ATML framework is defined by IEEE Std 1671™.^{1,2} IEEE Std 1671 is therefore a critical part of the ATML component defined by this standard.

This standard (and its associated schema) is intended to be used in identifying and documenting test descriptions that may be utilized during the testing of a particular unit under test (UUT).

1.2 Scope

The scope of this standard is the definition of an exchange format, utilizing eXtensible Markup Language (XML), for exchanging the test description information defining test performance, test conditions,

¹ The IEEE standards referenced in this standard are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

² For information on references, see Clause 2.

diagnostic requirements, and support equipment to locate, align, and verify the proper operation of a UUT. This is in support of the development of test program sets (TPSs) that will be used in an automatic test environment.

1.3 Purpose

The purpose of this standard is to promote and facilitate interoperability between components of Automatic Test Systems (ATSs) (e.g., rehosting test requirements between ATS platforms) where UUT requirement definitions need to be shared. This standard will also facilitate the preparation or documentation of test program(s). The TestDescription schema becomes a class of information that can be used within the ATML family of standards.

1.4 Application

The standard will permit test descriptions to be utilized for a variety of purposes, including test program generation, Test Requirement Document (TRD) development and maintenance (formerly supported by MIL-STD-1345B-1981 [B8]³ and MIL-STD-1519-1977 [B9]), and test description analysis. These test descriptions can reference IEEE Std 1641TM, which describes signals and their behavior.

The current test programs developed to run on ATSs are implemented with tight coupling between components. The components are typically developed specific to a particular architecture. Once the test program is fielded, the requirements and strategies used initially to develop the test program typically become obsolete as the test program evolves in its life cycle. As the ATS is replaced or achieves some level of obsolescence, it is typical to rehost the implementation of the TPS; rehosting is a more expensive, time-consuming task than that of implementing the UUT test requirement on the newer ATS or as part of instrument(s) replaced within the existing ATS. This standard will allow a common test description to be transportable across a variety of ATSs used in industries such as automotive, semiconductor, aerospace, and military.

The information contained in XML documents conforming to this standard will be useful to the following individuals:

- a) Test program developers
- b) Test program maintainers
- c) Automatic test system (ATS) developers
- d) ATS maintainers
- e) Developers of ATML-based tools and systems

1.5 Conventions used within this document

1.5.1 General

In accordance with the *IEEE Standards Style Manual* [B7], any XML schema examples (see XML Schema Tutorial [B10]) will be shown in `Courier` font. In cases where instance document examples are necessary

³ The numbers in brackets correspond to those of the bibliography in Annex E.

to depict the use of a schema type or element, such examples will also be shown in *Courier* font. When the characters “...” appear in an example, it indicates that the example component is incomplete.

All simple types, complex types, attribute groups, and elements will be listed; explanatory information will be provided, along with examples if additional clarification is needed. The explanatory information will include information on the intended use of the elements and/or attributes where the name of the entity does not clearly indicate its intended use. For elements derived from another source type (e.g., an abstract type), only attributes that extend the source type will be listed; details regarding the base type will be listed along with the base type.

When referring to an attribute of an XML element, the convention of *[element]@[attribute]* will be used. In cases where an attribute name is referred to with no associated element, the attribute name will be enclosed in single quotes. The names of keys and key references will also be enclosed in single quotes. Element and type names will always be set in italics when appearing in text.

In tables that describe XML attributes, the column “Use” indicates the occurrence constraints for each attribute.

- “Required” indicates that the attribute shall appear exactly once.
- “Optional” indicates that the attribute may appear once or not at all.

In tables that describe XML elements, the column “Use” indicates the occurrence constraints for each element.

- “Required” indicates that the element shall appear exactly once.
- “Optional” indicates that the attribute may appear once or not at all.
- “1..∞” indicates that the element shall appear at least once and may appear multiple times.
- “0..∞” indicates that the element may appear multiple times, once, or not at all.

This standard uses the vocabulary and definitions of relevant IEEE standards. In case of conflict of definitions, except for those portions quoted from standards, the following precedence shall be observed: 1) Clause 3, 2) Annex D, and 3) *The IEEE Standards Dictionary: Glossary of Terms & Definitions*.⁴

1.5.2 Precedence

The TestDescription schema (TestDescription.xsd) element, child element, and annotation information shall take precedence over the descriptive information contained in Clause 4.

The TestDescription schema and the material contained in Clause 4 shall take precedence over the TestDescription instance document information represented in Annex A as well as the examples in Annex B.

For clarity, portions of IEEE Std 1641 have been repeated within this standard. The current, approved version of IEEE Std 1641 shall take precedence over any IEEE Std 1641 material that is repeated within this standard.

1.5.3 Word usage

In accordance with *IEEE Standards Style Manual* [B7], the word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is

⁴ *The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at <http://shop.ieee.org/>.

permitted (*shall* equals *is required to*). The use of the word *must* is used only to describe unavoidable situations. The use of the word *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

The word *can* is used for statements of possibility and capability (*can* equals *is able to*).

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 1641™, IEEE Standard for Signal and Test Definition.^{5,6}

IEEE Std 1671™, IEEE Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Equipment and Test Information via XML.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms & Definitions* should be referenced for terms not defined in this clause. In the event a term is explicitly redefined, or defined in more detail in an ATML component standard, the component standards definition shall be normative for that ATML component standard.

abstract type: A declared type that can be used to define other types through derivation. Only nonabstract types derived from the declared type can be used in instance documents. When such a type is used, it must be identified by the 'xsi:type' attribute.

Automatic Test Markup Language (ATML) instance document: *See: instance document.*

element: A bounded component of the logical structure of an eXtensible Markup Language (XML) document that has a type and that may have eXtensible Markup Language (XML) attributes and content. (adapted from eXtensible Markup Language (XML) 1.0 (Fourth Edition) [B1])

entity: Something that has a distinct separate existence.

eXtensible Markup Language (XML) attribute: Name-value pair associated with an XML element.

⁵ IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (<http://standards.ieee.org/>).

⁶ The IEEE standards or products referred to in this clause are trademarks owned by the Institute of Electrical and Electronics Engineers, Incorporated.

eXtensible Markup Language (XML) document: A data object that conforms to the XML requirements for being well formed. In addition, the data object is valid if it additionally conforms to semantic rules of the XML schema.

eXtensible Markup Language (XML) schema: The definition of a class of XML document, typically expressed in terms of constraints on the structure and the content of documents of that class, above and beyond the basic syntax constraints imposed by XML itself.

instance document: An eXtensible Markup Language (XML) document that conforms to a particular XML schema.

object: An object consists of state and behavior. An object stores its states in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages).

object instance: A specific occurrence of an object.

well formed: Conforming to all eXtensible Markup Language (XML) syntax rules.

3.2 Acronyms and abbreviations

AC	alternating current
ATE	automatic test equipment
ATML	Automatic Test Markup Language
ATPG	automatic test program generation
ATS	automatic test system
BSC	basic signal component
DC	direct current
EQ	equal
GE	greater or equal
GT	greater than
LE	lower or equal
LT	lower than
NE	not equal
STD	signal and test definition
TPS	test program set
TSF	test signal framework
UTF-8	8-bit unicode transformation format
UUID	universally unique identifier
UUT	unit under test
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

4. Schema—TestDescription.xsd

4.1 General

Should the reader not have a general understanding of XML schemas or XML terminology, an XML Schema Tutorial [B10] is available on the World Wide Web. This tutorial will help with the general understanding of the contents of both this clause and the ATML Common schemas (Annex B of IEEE Std 1671), of which this clause makes reference.

In addition to the conventions specified in 1.5.1, the namespace prefix “c:” identifies that the type or attribute group is contained in B.1 of IEEE Std 1671 (Schema-Common.xsd) and the namespace prefix “hc:” identifies that the type or attribute group is contained in B.2 of IEEE Std 1671 (Schema-HardwareCommon.xsd).

4.2 Elements

This subclause contains the description of the root element of the schema and of its descendants.

4.2.1 Root (or document)

Exactly one element exists, called the *root*, or *document element*, of which no part appears in the content of any other element. This root element serves as the parent for all other elements of the TestDescription schema.

The TestDescription schema’s root element is defined as follows:

Name	Set to
Attribute form default	unqualified (see NOTE 1) ⁷
Element form default	qualified (see NOTE 1)
Encoding	8-bit unicode transformation format (UTF-8)
Included schema	<i>None</i>
Imported schema	urn:IEEE-1671:2009.02:Common urn:IEEE-1671:2009.02:HardwareCommon
Target namespace	urn:IEEE-1671.1:2009:TestDescription
Version	1.0
XML schema namespace reference	See NOTE 2
NOTE 1—Qualified and unqualified are described in A.3.6 of IEEE Std 1671.	
NOTE 2—This document is available from the World Wide Web Consortium (W3C) Web site http://www.w3.org/2001/XMLSchema .	

4.2.2 TestDescription

This element shall contain information defining test performance, test conditions, diagnostic requirements, and support equipment to locate, align, and verify the proper operation of a UUT.

Figure 1 illustrates the top-level XML complex types that comprise the TestDescription schema. Within Figure 1, “solid lined boxes” indicate that the XML type shall be required, whereas “dotted lined boxes” indicate that the XML type may be used. The referenced subclause identifies where the definition of the element is located within this standard.

⁷ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

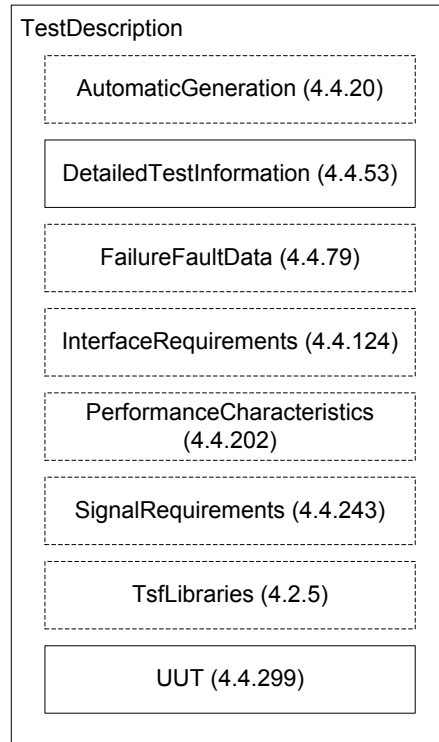


Figure 1—TestDescription element

4.2.2.1 Attributes

Name	Type	Description	Use
classified	<i>xs:Boolean</i>	This attribute shall be equal to “true” if the instance document is classified.	Optional
name	<i>c:NonBlankString</i>	Name of the instance document.	Optional
securityClassification	<i>c:NonBlankString</i>	A use-case determined string declaring the security classification level of the instance document.	Optional
uuid	<i>c:Uuid</i>	Universal unique identifier for the instance document.	Required
version	<i>c:NonBlankString</i>	Version of the instance document.	Optional

Attributes ‘classified,’ ‘uuid,’ and ‘securityClassification’ belong to the attribute group *c:DocumentRootAttributes* defined in B.1 of IEEE Std 1671.

4.2.2.2 Child elements

Name	Type	Description	Use
AutomaticGeneration	<i>AutomaticGeneration</i> , see 4.4.20	Shall identify the Automatic Test Program Generation (ATPG) tools used to generate the test program.	Optional
DetailedTestInformation	<i>DetailedTestInformation</i> , see 4.4.53	Shall contain all of the sufficient data for each UUT test to describe completely all input conditions and measurements required.	Required
FailureFaultData	<i>FaultData</i> , see 4.4.79	Shall contain UUT fault data, including faults and failures.	Optional

Name	Type	Description	Use
		The element shall not appear in instance documents unless it has at least one child element. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.	
InterfaceRequirements	<i>InterfaceRequirements</i> , see 4.4.124	Shall describe the characteristics of equipment and circuitry required to test the UUT, excluding the test equipment (e.g., UUT connector information).	Optional
PerformanceCharacteristics	<i>PerformanceCharacteristics</i> , see 4.4.202	Shall describe the performance characteristics of the UUT. The element shall not appear in instance documents unless it has at least one child element. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.	Optional
SignalRequirements	<i>SignalRequirements</i> , see 4.4.243	Shall specify all UUT signal requirements for testing the UUT.	Optional
TsfLibraries	Type defined inline	See 4.2.5.	Optional
UUT	<i>UUT</i> , see 4.4.299	Shall contain descriptive information for the UUT.	Required

4.2.2.3 Constraints

Name	Type	Refers	Selector	Fields
componentKey	<i>Key</i>		//Component	@ID
componentRef	<i>Keyref</i>	componentKey	//AdjustComponent	@componentID
controlKey	<i>Key</i>		//PerformanceCharacteristics/Controls/Control	@ID
controlRef	<i>Keyref</i>	controlKey	//RelatedControls/RelatedControl	@controlID
failureKey	<i>Key</i>		//FailureFaultData/Failures/Failure	@ID
failureNextStepRef	<i>Keyref</i>	failureKey	//NextStep/DetectionIsolation/Failures/Failure	@failureID
failureOutcomeRef	<i>Keyref</i>	failureKey	//Outcome/DetectionIsolation/Failures/Failure	@failureID
faultKey	<i>Key</i>		//Component/Faults/Fault	@ID
faultNextStepRef	<i>Keyref</i>	faultKey	//NextStep/DetectionIsolation/Faults/Fault	@faultID
faultOutcomeRef	<i>Keyref</i>	faultKey	//Outcome/DetectionIsolation/Faults/Fault	@faultID
inputKey	<i>Key</i>		//PerformanceCharacteristics/Inputs/Input	@ID
inputRef	<i>Keyref</i>	inputKey	//RelatedInputs/RelatedInput	@inputID
toolKey	<i>Key</i>		//Tools/Tool	@ID
toolRef	<i>Keyref</i>	toolKey	//td:Behavior/td:AutomaticGeneration	@toolID
tsfLibraryKey	<i>Key</i>		//TsfLibraries/TsfLibrary	@ID
tsfLibraryRef	<i>Keyref</i>	tsfLibraryKey	//TsfClass	@tsfLibraryID

4.2.3 TestDescription/DetailedTestInformation

Shall contain all of the sufficient data for each UUT test to completely describe all input conditions and measurements required.

NOTE—This subclause describes the constraints associated with the element. Refer to 4.4.53 for a description of the element's type.

4.2.3.1 Attributes

Refer to 4.4.53.1 for a description of the element's attributes.

4.2.3.2 Child elements

Refer to 4.4.53.2 for a description of this element's child elements.

4.2.3.3 Constraints

Name	Type	Refs	Selector	Fields
actionKey	Key		//Action	@ID
actionInitializationRef	Keyref	actionKey	//InitializationAction	@actionID
actionTerminationRef	Keyref	actionKey	//TerminationAction	@actionID
actionStepRef	Keyref	actionKey	//Step/Action Reference	@actionID
actionTestGroupRef	Keyref	actionKey	//TestGroup/ActionRefs/Action Reference	@actionID
actionEntryPointRef	Keyref	actionKey	//ActionEntryPoint	@actionID
actionOutcomeKey	Key		//Action/Outcomes/Outcome	@ID
actionOutcomeResultRef	Keyref	actionOutcomeKey	//Result/ActionOutcomeReference	@actionOutcomeID
actionOutcomeOperationRef	Keyref	actionOutcomeKey	//Operation/Results/Result	@actionOutcomeID
globalSignalKey	Key		//GlobalSignal/Source	@ID
globalSignalReferenceRef	Keyref	globalSignalKey	//GlobalSignalReference	@globalSignalID
globalSignalOperationRef	Keyref	globalSignalKey	//td:GlobalSignalOperation	@globalSignalID
measurementKey	Key		//Operation/Measurement	@ID
measurementRef	Keyref	operationMeasurementKey	//c:Datum	@measurementID
sessionDatumKey	Key		//Action/SessionData/SessionDatum	@ID
sessionDatumDatumRef	Keyref	sessionDatumKey	//c:Datum/SessionDatum	@sessionDatumID
sessionDatumOperationRef	Keyref	sessionDatumKey	//Operation/ValueToSessionDatum	@sessionDatumID
stateVariableKey	Key		//StateVariable	@ID
stateVariableOperationRef	Keyref	stateVariableKey	//Operation	@stateVariableID
stateVariableValueKey	Key		//StateVariable/Values/Value	@ID
stateVariableValueInitialRef	Keyref	stateVariableValueKey	//StateVariable/InitialValue	@valueID
stateVariableValuePreConditionRef	Keyref	stateVariableValueKey	//PreCondition/StateVariableValue	@stateVariableValueID
stateVariableValuePostConditionRef	Keyref	stateVariableValueKey	//PostCondition/StateVariableValue	@stateVariableValueID
stateVariableValueOperationRef	Keyref	stateVariableValueKey	//Operation	@stateVariableValueID
testGroupKey	Key		//TestGroup	@ID
testGroupEntryPointRef	Keyref	testGroupKey	//TestGroupEntryPoint	@testGroupID
testGroupCallRef	Keyref	testGroupKey	//Behavior/TestGroupCall	@testGroupID
testGroupStepRef	Keyref	testGroupKey	//Step/TestGroupReference	@testGroupID
testGroupInitializationRef	Keyref	testGroupKey	//InitializationTestGroup	@testGroupID
testGroupTerminationRef	Keyref	testGroupKey	//TerminationTestGroup	@testGroupID
testGroupOutcomeKey	Key		//TestGroup/Outcomes/Outcome	@ID

Name	Type	Refers	Selector	Fields
testGroupOutcomeRef	<i>Keyref</i>	testGroupOutcomeKey	./Step/Results/Result/TestGroupOutcomeReference	@testGroupOutcomeID
testGroupParameterKey	<i>Key</i>		./TestGroup/ParameterDescriptions/ParameterDescription	@ID
testGroupParameterDatumRef	<i>Keyref</i>	testGroupParameterKey	./c:Datum	@testGroupParameterID
testGroupParameterValueRef	<i>Keyref</i>	testGroupParameterKey	./ValueToParameter	@testGroupParameterID
testGroupTestResultKey	<i>Key</i>		./TestGroup/TestResultDescriptions/TestResultDescription	@ID
testGroupTestResultTestResultRef	<i>Keyref</i>	testGroupTestResultKey	./Action/TestResults/TestResult/ValueToTestResult	@testGroupTestResultID
testGroupTestResultSessionDatumRef	<i>Keyref</i>	testGroupTestResultKey	./Action/SessionData/SessionDatum/ValueToTestResult	@testGroupTestResultID
testResultKey	<i>Key</i>		./Action/TestResults/TestResult	@ID
testResultDatumRef	<i>Keyref</i>	testResultKey	./c:Datum/TestResult	@testResultID
testResultOperationRef	<i>Keyref</i>	testResultKey	./Operation/ValueToTestResult	@testResultID
testResultOutValueRef	<i>Keyref</i>	testResultKey	./IeeeStd1641/IeeeStd1641OutValues/ResultOutValue	@testResultID

4.2.4 TestDescription/InterfaceRequirements

TestDescription/InterfaceRequirements shall describe the characteristics of equipment and circuitry required to test the UUT, excluding the test equipment (e.g., UUT connector information).

NOTE—This section describes the constraints associated with the element. Refer to 4.4.124 for a description of the element's type.

4.2.4.1 Attributes

Refer to 4.4.124.1 for a description of the element's attributes.

4.2.4.2 Child elements

Refer to 4.4.124.2 for a description of this element's child elements.

4.2.4.3 Constraints

Name	Type	Refers	Selector	Fields
interfaceConnectorKey	<i>Key</i>		./c:Connectors/c:Connector	@ID
interfaceConnectorRef	<i>Keyref</i>	interfaceConnectorKey	./c:Ports/c:Port/c:ConnectorPins/c:ConnectorPin	@connectorID
interfacePortNamesUnique	<i>Unique</i>		./c:Ports/c:Port	@name
testPointKey	<i>Key</i>		./TestPoints/TestPoint	@ID
electroOpticalInterfaceKey	<i>Key</i>		./ElectroOpticalInterfaces/ElectroOpticalInterface	@ID

4.2.5 TestDescription/TsfLibraries

Shall identify the test signal framework (TSF) libraries referenced in the current instance document. These TSF libraries describe the interface and behavior of IEEE 1641 signals used in the instance document.

A TSF library can be described through an XML schema, which describes the interface of signals (i.e., their attributes), or by an XML instance document, which describes the interface of signals and defines their behavior. Refer to Clause 6 of IEEE Std 1641 for information on describing TSF libraries.

The XML schemas of all libraries referenced by a TestDescription instance document will be referenced from the root element of that instance document, allowing the validation of IEEE 1641 signals the document defines. The child elements of *TsfLibraries* shall reference both the XML schemas and the XML instance documents of the TSF libraries, allowing consumers to determine the interface and the behavior of signals from those libraries.

4.2.5.1 Attributes

This element definition contains no attributes.

4.2.5.2 Child elements

Name	Type	Description	Use
TsfLibrary	Type defined inline	See 4.2.6.	1..∞

4.2.6 TestDescription/TsfLibraries/TsfLibrary

Identifies one TSF library by referencing the external XML documents that describe the interface and behavior of signals from the TSF library.

4.2.6.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing of the <i>TsfLibrary</i> element.	Required
name	<i>c:NonBlankString</i>	Descriptive name for the TSF library.	Optional

4.2.6.2 Child elements

Name	Type	Description	Use
XmlInstanceDocumentURL	<i>c:NonBlankURI</i>	References the XML schema document that defines the interface of the TSF library.	Required
XmlSchemaURL	<i>c:NonBlankURI</i>	References the XML instance document that describes the interface and behavior of signals from the TSF library. For each TSF library, the file referenced by this element shall be identical to the file referenced by the 'xsi:schemaLocation' attribute of the root element. Note that the previous constraint is not enforced by this XML schema. Consumer applications should verify the constraint programmatically.	Required

4.3 Simple types

4.3.1 ComparisonResult

Base type: restriction of *c:string*

Enumerations: Equal | NotEqual | InTolerance | FailHigh | FailLow

Enumeration of possible values for comparison results.

4.3.2 Detectability

Base type: restriction of *xs:NMTOKENS*

Enumerations: DetectableInsertable | DetectableDestructive | NonDetectable

Enumeration of fault types, with respect to detectability, as follows:

- a) DetectableInsertable: The fault can be inserted into the UUT and detected by functional tests at the channel, division, or system level; the performance of the functional tests will cause no adverse effect on the life cycle of the UUT.
- b) DetectableDestructive: The fault could be inserted into the UUT and detected by functional tests; however, the execution of the functional tests may adversely affect the life expectancy of the UUT or render the UUT unfit for use.
- c) NonDetectable: Fault identified by analysis that cannot be detected by a functional test or tests.

4.3.3 ElectroOpticalInterfaceType

Base type: restriction of *xs:NMTOKENS*

Enumerations: Laser | Infrared | Light | Other

Enumeration of types for electro-optical interfaces.

4.3.4 FailureType

Base type: restriction of *xs:NMTOKENS*

Enumerations: Detectable | NonDetectable | NonFunctional

Enumeration of failure types, as follows:

- a) Detectable: The failure can be detected by a functional test.
- b) NonDetectable: The failure cannot be detected by any functional test.
- c) NonFunctional: The failure can be detected by a functional test, but no repair action is required because the failure will not affect any higher assembly in any operating mode under specified service environmental conditions.

4.3.5 OutcomeValue

Base type: restriction of *c:string*

Enumerations: Passed | Failed | Aborted

Enumeration of possible values for outcomes of tests and test groups.

4.3.6 SeverityLevel

Base type: restriction of *xs:NMTOKENS*

Enumerations: Warning | Error | Fatal

Enumeration of severity levels for faults and failures, as follows:

- a) Warning: The fault or failure may impact UUT functionality and has no long-term impact on UUT reliability.
- b) Error: The fault or failure impacts UUT functionality and impacts UUT reliability (i.e., life is shortened).
- c) Fatal: The UUT is unusable.

4.3.7 SignalCharacteristicRole

Base type: restriction of *c:string*

Enumerations: Control | Limit | Measurement | Capability

Enumeration of signal characteristic roles, as follows:

- a) Control: The characteristic is controlled by an automatic test equipment (ATE) resource.
- b) Limit: The characteristic is limited by an ATE resource.
- c) Measurement: The characteristic is measured by an ATE resource.
- d) Capability: The characteristic is related to the capacity of an ATE resource to satisfy a test requirement, but it is not required to be controlled, limited, or measured.

4.3.8 SignalRole

Base type: restriction of *c:string*

Enumerations: Source | Sensor | Monitor | Load

Enumeration of signal roles, as follows:

- a) Source: The signal is generated by an ATE resource.
- b) Sensor: The signal is measured by an ATE resource.
- c) Monitor: The signal is monitored by an ATE resource.
- d) Load: An ATE resource applies a load to the UUT.

4.3.9 SignalType

Base type: restriction of *xs:NMTOKENS*

Enumerations: Input | Output | Bi-Directional | Load

Enumeration of signal types, as follows:

- a) Input: The signal is received by the UUT.
- b) Output: The signal is generated by the UUT.
- c) Bi-Directional: The signal is both received and generated by the UUT.
- d) Load: The signal describes a load applied to the UUT.

4.3.10 TestType

Base type: restriction of *c:string*

Enumerations: UutIdentification | PowerStimuliShort | SafeToTurnOn | Performance | Diagnostic

Enumeration of test types, as follows:

- a) UutIdentification: Verifies that the model of the current UUT is compatible with the testing procedure described in the instance document.
- b) PowerStimuliShort: Verifies power or stimulus inputs to be not shorted prior to any stimulus application to the UUT.
- c) SafeToTurnOn: Verifies that it is safe to turn on power.
- d) Performance: Verifies proper operation of the UUT.
- e) Diagnostic: Isolates faults within the UUT.

4.4 Complex types

4.4.1 ActionOutcomeType (abstract)

Abstract base type for describing the outcome of an action (test or session action) or test group. The following derived types are defined in the schema:

- a) *Outcome*, see 4.4.201
- b) *SessionOutcome*, see 4.4.241

Note that this is an abstract type that is used as base type for other type definitions. Where this type is assigned in the schema, elements with derived types shall be provided. The 'xsi:type' attribute shall be used to indicate the type of these elements.

4.4.1.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the action outcome.	Required
qualifier	<i>c:NonBlankString</i>	Outcome qualifier (e.g., “High” and “Low”). When used, shall contain an adjective providing additional descriptive data for the ‘value’ attribute defined in derived types. When the ‘value’ attribute (defined on each of the derived types) contains “Aborted,” the ‘qualifier’ attribute shall provide essential descriptive or explanatory information regarding the reason for the action or test group not completing normally. Refer to B.2.12 for an example using qualifiers.	Optional

4.4.1.2 Child elements

Name	Type	Description	Use
Description	<i>c:NonBlankString</i>	Free-form description of the outcome.	Optional
Extension	<i>c:Extension</i>	Provided to permit schema extensibility.	Optional
PostConditions	<i>PostConditions</i> , see 4.4.218	Shall describe all postconditions for the action or test group, or for an individual outcome of that test or test group. Postconditions defined in a child element of <i>Test/Conditions</i> or <i>SessionAction/Condition</i> shall apply to all outcomes of that test or session action. Postconditions defined in a child element of <i>Outcome</i> shall apply only for that particular outcome, overriding the postconditions defined in a child element of <i>Test/Conditions</i> or <i>SessionAction/Condition</i> . The preconditions can be fulfilled by executing actions or test groups with the appropriate postconditions. For example, a session action can apply power, download firmware, or change the temperature of a thermal chamber; a test can verify the signature of the UUT, and so on. Note that postconditions can be specified for both actions (i.e., tests or session outcomes) and action outcomes. The postconditions specified for actions become fulfilled after action execution, regardless of the outcome produced by that action. The postconditions specified for action outcomes become fulfilled after action execution only if that specific outcome was produced by the action. If the postconditions are specified at both levels, then the postconditions associated with action outcomes override those associated with the action.	Optional

4.4.2 ActionResultType (abstract)

Abstract base type for describing one result of an action (i.e., test or session action).

Note that this type is used as base type for other type definitions. The following elements with derived types are defined inline in the schema:

- a) *Test/TestResults/TestResult*, see 4.4.263
- b) *SessionAction/SessionData/SessionDatum*, see 4.4.240

Refer to B.2.18 for usage information regarding action results.

4.4.2.1 Attributes

Name	Type	Description	Use
documentRequirementID	<i>c:NonBlankString</i>	Reference to a requirement ID from an external requirements document, which in turn shall be referenced by a <i>Document</i> element under <i>Documentation/ Requirements</i> . Note that the format and location of requirement IDs in the external document are not enforced by this standard and can be application specific.	Optional
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the result.	Required
name	<i>c:NonBlankString</i>	Name of the action result.	Required

4.4.2.2 Child elements

Name	Type	Description	Use
Description	<i>c:NonBlankString</i>	Free-form description of the result.	Optional
Extension	<i>c:Extension</i>	Provided to permit schema extensibility.	Optional
ValueDescription	Type defined inline	See 4.4.3	Required
ValueToTestResult	Type defined inline	See 4.4.4.	Optional

4.4.3 ActionResultType/ValueDescription

Shall describe the data type and the unit of the result.

Base type: extension of *ValueDescription*, see 4.4.310.

Types derived from *DatumDescriptionType* shall be assigned to *DatumDescription* descendants. Resolution, ErrorLimit, Range, and Confidence information can be provided under *DatumQuality* descendants.

4.4.3.1 Attributes

This element definition contains no attributes.

4.4.3.2 Child elements

This element definition contains no child elements.

4.4.4 ActionResultType/ValueToTestResult

When present, this element shall indicate that after action execution, the value of the current action result is transferred to a test result of the containing test group.

The data type, unit, and qualifier of the referenced test result shall be identical to those of the current action result. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

4.4.4.1 Attributes

Name	Type	Description	Use
testGroupTestResultID	<i>c:NonBlankString</i>	ID of the test result the value of the current action result is transferred to.	Required

4.4.4.2 Child elements

This element definition contains no child elements.

4.4.5 ActionType (abstract)

Abstract base type for test actions. The following derived types are defined in the schema:

- a) *Test*, see 4.4.260
- b) *SessionAction*, see 4.4.238

Note that this is an abstract type, used as base type for other type definitions. Where this type is assigned in the schema, elements with derived types shall be provided. The ‘xsi:type’ attribute shall be used to indicate the type of these elements.

4.4.5.1 Attributes

Name	Type	Description	Use
classified	<i>xs:boolean</i>	Indicates if the action is classified.	Optional
cost	<i>xs:double</i>	Cost of action execution, representing the duration of execution, the financial cost associated with execution, or a value calculated by an objective function. May be used to optimize the order of execution.	Optional
customType	<i>c:NonBlankString</i>	Custom action type, used to indicate actions that belong to user-defined categories. The semantics attached to categories are user specific. For example, actions from the same category may be implemented as a single function or procedure. When this attribute is used to provide a guideline for implementation, implementers may choose to follow or ignore the guideline.	Optional
documentRequirementID	<i>c:NonBlankString</i>	Reference to a requirement ID from an external requirements document, which in turn shall be referenced by a <i>Document</i> element under <i>Documentation/Requirements</i> . Note that the format and location of requirement IDs in the external document are not mandated by this	Optional

Name	Type	Description	Use
		standard and can be application specific.	
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the action.	Required
name	<i>c:NonBlankString</i>	Name of the action.	Required
securityClassification	<i>c:NonBlankString</i>	User-defined security classification for the action.	Optional

4.4.5.2 Child elements

Name	Type	Description	Use
Behavior	Type defined inline	See 4.4.6.	Required
Conditions	Type defined inline	See 4.4.11.	Optional
Description	<i>c:NonBlankString</i>	Free-form description for the action.	Optional
Extension	<i>c:Extension</i>	Provided to permit schema extensibility.	Optional
LocalSignals	Type defined inline	See 4.4.12.	Optional
Parameters	Type defined inline	See 4.4.17.	Optional

4.4.6 ActionType/Behavior

Shall describe the behavior of the action.

Refer to B.2.14 for usage information regarding behavior.

4.4.6.1 Attributes

This element definition contains no attributes.

4.4.6.2 Child elements

	Name	Type	Description	Use
Choice	AutomaticGeneration	Type defined inline	See 4.4.7.	Required
	Description	<i>c:NonBlankString</i>	The presence of this element shall indicate that behavior is specified using free-form text. The element shall contain the descriptive text. Refer to B.2 for an example that contains free-form description of test behavior.	
	Extension	<i>c:Extension</i>	Provided to permit schema extensibility. Through this element, test behavior may be specified using XML content that conforms to an external XML schema.	
	IeeeStd1641	<i>IeeeStd1641</i>	The presence of this element shall indicate that behavior is specified as a test definition conformant to IEEE Std 1641. Refer to B.3 for an example where test behavior is described through IEEE 1641 signals. The element shall not appear in instance documents unless it has at least one child element. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.	
	Operations	<i>Operations</i>	The presence of this element indicates that behavior is specified as a sequence of predefined operations. The operation described by the child elements shall be executed sequentially, in the order in which they appear in the instance document. Refer to B.4 for an example where test behavior is described through operations.	
	TestGroupCall	Type defined inline	See 4.4.10.	
NOTE—Elements <i>AutomaticGeneration</i> , <i>Description</i> , <i>Extension</i> , <i>IeeeStd1641</i> , <i>Operations</i> , and <i>TestGroupCall</i> are alternative elements.				

4.4.7 ActionType/Behavior/AutomaticGeneration

The presence of this element shall indicate that the part of the test program that corresponds to the current action will be generated automatically by an ATPG tool. The child elements shall identify the tool and the data files used for generation.

4.4.7.1 Attributes

Name	Type	Description	Use
toolID	<i>c:NonBlankString</i>	ID of the element describing the ATPG tool that generates the test.	Required

4.4.7.2 Child elements

Name	Type	Description	Use
DataFiles	Type defined inline	See 4.4.8.	Optional

4.4.8 ActionType/Behavior/AutomaticGeneration/DataFiles

Shall contain the list of all data files processed or generated by the ATPG tool for the current test.

4.4.8.1 Attributes

This element definition contains no attributes.

4.4.8.2 Child elements

Name	Type	Description	Use
DataFile	<i>c:Document</i>	Shall identify one data file.	1..∞

4.4.9 ActionType/Behavior/IeeeStd1641

The presence of this element shall indicate that behavior is specified as a test definition conformant to IEEE Std 1641. Refer to B.3 for an example where test behavior is described through IEEE 1641 signals.

The element shall not appear in instance documents unless it has at least one child element. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

NOTE—This subclause describes the constraints associated with the element. Refer to 4.4.96 for a description of the element's type.

4.4.9.1 Attributes

Refer to 4.4.96.1 for a description of element's attributes.

4.4.9.2 Child elements

Refer to 4.4.96.2 for a description of this element's child elements.

4.4.9.3 Constraints

Name	Type	Refers	Selector	Fields
signalNameKey	<i>Key</i>		<i>.//*</i>	<i>@name</i>
signalNameRef	<i>Keyref</i>	signalNameKey	<i>.//*</i>	<i>@signalName</i>

4.4.10 ActionType/Behavior/TestGroupCall

The presence of this element shall indicate that the action consists in the execution of a test group. The element shall identify the test group.

If the test group has multiple entry points, then execution shall start at the primary entry point.

When the behavior specified for an action is “test group call,” the outcomes defined for that action shall be identical to those of the referenced test group. Note that the previous constraint is not enforced by this XML schema. Consumer applications should verify the constraint programmatically. Refer to B.4 for an example that contains test group calls.

4.4.10.1 Attributes

Name	Type	Description	Use
testGroupID	<i>c:NonBlankString</i>	ID of the test group that is executed.	Required

4.4.10.2 Child elements

This element definition contains no child elements.

4.4.11 ActionType/Conditions

Shall specify the preconditions and postconditions for the action.

The element shall not appear in instance documents unless it has at least one child element. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

4.4.11.1 Attributes

This element definition contains no attributes.

4.4.11.2 Child elements

Name	Type	Description	Use
PostConditions	<i>PostConditions</i> , see 4.4.218	Shall describe all postconditions for the action or test group. Preconditions can be fulfilled by executing actions or test groups with the appropriate postconditions. For example, a session action can apply power, download firmware, or change the temperature of a thermal chamber; a test can verify the signature of the UUT, and so on. Note that postconditions can be specified for both actions and action outcomes. The postconditions specified for actions become fulfilled after action execution, regardless of the outcome produced by that action. The postconditions specified for action outcomes become fulfilled after action execution only if that specific outcome was produced by the action. If the postconditions are specified at both levels, then the postconditions associated with action outcomes override those associated with the action.	Optional
PreConditions	<i>PreConditions</i> , see 4.4.221	Shall describe all preconditions for the action or test group. The specification of preconditions aims to ensure that the actions and test groups are executed only if the proper conditions exist. For example, power is applied to the UUT, the appropriate firmware is loaded in the UUT, the UUT temperature is within specified limits, and so on.	Optional

4.4.12 ActionType/LocalSignals

Shall identify local signals, shared by multiple operations specified for the current action. Local signals are referenced from operations. Refer to B.4.2 for usage information regarding local signals and to B.4 for an example.

4.4.12.1 Attributes

This element definition contains no attributes.

4.4.12.2 Child elements

Name	Type	Description	Use
LocalSignal	Type defined inline	See 4.4.13.	1..∞

4.4.13 ActionType/LocalSignals/LocalSignal

Shall describes one local signal, which may be a source signal, a measurement, or a connection.

4.4.13.1 Attributes

Name	Type	Description	Use
Name	<i>c:NonBlankString</i>	Name of the local signal. May be used for documentation purposes.	Optional

4.4.13.2 Child elements

	Name	Type	Description	Use
Choice	Monitor	Type defined inline	See 4.4.14.	Required
	Sensor	Type defined inline	See 4.4.15.	
	Source	Type defined inline	See 4.4.16.	
NOTE—Elements <i>Monitor</i> , <i>Sensor</i> , and <i>Source</i> are alternative elements.				

4.4.14 ActionType/LocalSignals/LocalSignal/Monitor

Shall describe a local monitor signal. Can be referenced from operations with type *OperationSetup*, *OperationChange*, *OperationConnect*, *OperationEnable*, *OperationDisable*, *OperationDisconnect*, and *OperationReset* located in the current action. Can also be referenced from *Synchronization* child elements of any *Operation* element located in the current action.

4.4.14.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the local monitor signal.	Optional

4.4.14.2 Child elements

This element definition contains no child elements.

4.4.15 ActionType/LocalSignals/LocalSignal/Sensor

Shall describe a local sensor signal. Can be referenced from operations with type *OperationSetup*, *OperationChange*, *OperationConnect*, *OperationRead*, *OperationDisconnect*, and *OperationReset* located in the current action.

4.4.15.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the local sensor signal.	Optional

4.4.15.2 Child elements

This element definition contains no child elements.

4.4.16 ActionType/LocalSignals/LocalSignal/Source

Shall describe a local source signal. Can be referenced from operations with type *OperationSetup*, *OperationChange*, *OperationConnect*, *OperationDisconnect*, and *OperationReset* located in the current action.

4.4.16.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the local source signal.	Optional

4.4.16.2 Child elements

This element definition contains no child elements.

4.4.17 ActionType/Parameters

Shall describe the parameters of the action and shall specify their values. Refer to Annex B for examples on describing parameters. Refer to B.2.15 for usage information regarding parameters.

4.4.17.1 Attributes

This element definition contains no attributes.

4.4.17.2 Child elements

Name	Type	Description	Use
Parameter	Type defined inline	See 4.4.18.	1..∞

4.4.18 ActionType/Parameters/Parameter

Shall describe one parameter of the action and shall specify its value.

Parameters shall be used to represent values that determine the behavior of the action. This includes, for example, the characteristics of stimulus signals, measurement conditions, UUT pins where signals are applied or measured, the limits of parametric sweeps, the indices of UUT channels to be tested, string commands to be sent to the UUT, identification of firmware to be loaded in the UUT, and so on.

Parameters shall not be used to represent the limits of acceptable UUT operation, as those limits are described using *TestResult* descendants of complex type *Test*, which is derived from *ActionType*.

4.4.18.1 Attributes

Name	Type	Description	Use
documentRequirementID	<i>c:NonBlankString</i>	Reference to a requirement ID from an external requirements document, which in turn shall be referenced by a <i>Document</i> element under <i>Documentation/Requirements</i> . Note that the format and location of requirement IDs in the external document are not enforced by this standard and can be application specific.	Optional
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the parameter. Note that constraints require parameter IDs to be unique only within their containing action; consequently, parameters cannot be referenced from outside that action.	Required
Name	<i>c:NonBlankString</i>	Name of the parameter.	Required

4.4.18.2 Child elements

Name		Type	Description	Use
Description		<i>c:NonBlankString</i>	Free-form description of the parameter.	Optional
Extension		<i>c:Extension</i>	Provided to permit schema extensibility.	Optional
ValueToParameter		Type defined inline	See 4.4.19.	Optional
Choice	Reference	<i>c:Document</i>	When present, shall indicate that the value will be retrieved at run time from an external document, instead of being encoded in the test program. This element shall reference the external document.	Required
	Value	<i>c:Value</i>	Shall specify the value of the parameter. The attributes of this element shall specify the unit (when applicable) and may specify a qualifier. Resolution, ErrorLimit, Range, and Confidence information can be provided under <i>DatumQuality</i> descendants. The type <i>ConnectionDatum</i> shall be assigned to <i>Datum</i> descendants of this element to specify a connection as the parameter value. The type <i>ValueFromTestGroupParameter</i> shall be assigned to <i>Datum</i> descendants of this element to indicate a parameter value that originates from a parameter of the containing test group. The type <i>ValueFromActionResults</i> shall be assigned to <i>Datum</i> descendants of this element to indicate a parameter value that originates from one or more test results of tests or session data of session actions from the same test group.	
NOTE—Elements <i>Value</i> and <i>Reference</i> are alternative elements.				

4.4.19 ActionType/Parameters/Parameter/ValueToParameter

When present, shall indicate that before action execution, the value of the current parameter shall be assigned to a parameter of the called test group.

The element can be present only when the current parameter belongs to an action whose behavior is “TestGroupCall.”

The data type, unit, and qualifier of the referenced test group parameter shall be identical to those of the current parameter. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

4.4.19.1 Attributes

Name	Type	Description	Use
testGroupParameterID	<i>c:NonBlankString</i>	ID of the test group parameter.	Required

4.4.19.2 Child elements

This element definition contains no child elements.

4.4.20 AutomaticGeneration

Shall define the ATPG tools that may be used to generate the test program or parts of the test program. The following are examples of ATPG tools: UUT simulators, fault simulators, test pattern generators, translators, postprocessors, and so on.

4.4.20.1 Attributes

This type definition contains no attributes.

4.4.20.2 Child elements

Name	Type	Description	Use
Extension	<i>c:Extension</i>	User-defined ATPG data.	Optional
Tools	Type defined inline	See 4.4.21.	Required

4.4.21 AutomaticGeneration/Tools

Shall contain the list of ATPG tools.

4.4.21.1 Attributes

This element definition contains no attributes.

4.4.21.2 Child elements

Name	Type	Description	Use
Tool	Type defined inline	See 4.4.22.	1..∞

4.4.22 AutomaticGeneration/Tools/Tool

Shall describe one ATPG tool.

Base type: extension of *c:SoftwareInstance*

4.4.22.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	Unique identifier, enables referencing the ATPG tool.	Required
name	<i>c:NonBlankString</i>	Optional tool name.	Optional
type	<i>c:NonBlankString</i>	Tool type (e.g., pattern generator).	Required

4.4.22.2 Child elements

Name	Type	Description	Use
DataFormat	Type defined inline	See 4.4.23.	Optional
Extension	<i>c:Extension</i>	Provided to permit schema extensibility.	Optional

4.4.23 AutomaticGeneration/Tools/Tool/DataFormat

Shall describe the format of data processed by the ATPG tool (e.g., IEEE Std 1445™-1998 [B4]).

4.4.23.1 Attributes

Name	Type	Description	Use
name	<i>c:NonBlankString</i>	Name of the format.	Optional

4.4.23.2 Child elements

Name	Type	Description	Use
ComplianceDocument	<i>c:Document</i>	References a compliance document applicable to the format (e.g., IEEE Std 1445-1998 [B4]).	0..∞
Description	<i>c:NonBlankString</i>	Free-form description of the format.	Optional

4.4.24 binaryArrayDescription

Base type: extension of *IndexedArrayDescriptionType*, see 4.4.116

Describes an array of binary values. Where applicable, can be assigned to elements of type *IndexedArrayDescriptionType*.

Note that it is possible to have elements with this type that have no elements or attributes. The presence of such an element indicates that the described entity is an array with unknown size and number of dimensions where elements have no unit, that a default value for elements is not specified, and that a nominal value for the described entity is not specified.

4.4.24.1 Attributes

This type definition contains no attributes.

4.4.24.2 Child elements

Name	Type	Description	Use
DefaultElementValue	<i>c:binary</i>	Default value, applies to all array elements whose value is not explicitly defined through <i>NominalElementValue</i> child elements.	Optional
NominalElementValue	Type defined inline	See 4.4.25.	0..∞

4.4.25 binaryArrayDescription/NominalElementValue

The value of an array element, for the nominal array value.

Base type: extension of *c:binary*

4.4.25.1 Attributes

Name	Type	Description	Use
position	<i>c:ArrayIndexor</i>	Index of the element.	Required

4.4.25.2 Child elements

This element definition contains no child elements.

4.4.26 binaryDescription

Base type: extension of *DatumDescriptionType*, see 4.4.48

Describes a binary value. Where applicable, can be assigned to elements of type *DatumDescriptionType*.

4.4.26.1 Attributes

This element definition contains no attributes.

4.4.26.2 Child elements

Name	Type	Description	Use
NominalValue	<i>c:binary</i>	Nominal value.	Optional

4.4.27 booleanArrayDescription

Base type: extension of *IndexedArrayDescriptionType*, see 4.4.116

Describes an array of boolean values. Where applicable, can be assigned to elements of type *IndexedArrayDescriptionType*.

Note that it is possible to have elements with this type that have no elements or attributes. The presence of such an element indicates that the described entity is an array with unknown size and number of dimensions, where elements have no unit, that a default value for elements is not specified, and that a nominal value for the described entity is not specified.

4.4.27.1 Attributes

This type definition contains no attributes.

4.4.27.2 Child elements

Name	Type	Description	Use
DefaultElementValue	<i>c:boolean</i>	Default value, applies to all array elements whose value is not explicitly defined through <i>NominalElementValue</i> child elements.	Optional
NominalElementValue	Type defined inline	See 4.4.28.	0..∞

4.4.28 booleanArrayDescription/NominalElementValue

The value of an array element, for the nominal array value.

Base type: extension of *c:boolean*

4.4.28.1 Attributes

Name	Type	Description	Use
position	<i>c:ArrayIndexor</i>	Index of the element.	Required

4.4.28.2 Child elements

This element definition contains no child elements.

4.4.29 booleanDescription

Base type: extension of *DatumDescriptionType*, see 4.4.48

Describes a boolean value. Where applicable, can be assigned to elements of type *DatumDescriptionType*.

4.4.29.1 Attributes

This type definition contains no attributes.

4.4.29.2 Child elements

Name	Type	Description	Use
NominalValue	<i>c:boolean</i>	Nominal value.	Optional

4.4.30 CollectionDescription

Describes a value that consists of a set of heterogeneous data items.

4.4.30.1 Attributes

Name	Type	Description	Use
nonStandardUnit	<i>c:NonBlankString</i>	Default nonstandard unit, applies to data items for which a unit is not explicitly defined. This attribute supports any nonstandard unit not already defined in IEEE Std 260.1™-2004 [B2].	Optional
standardUnit	<i>c:StandardUnit</i>	Default standard unit, applies to data items for which a unit is not explicitly defined. When used, this attribute shall only contain a unit of measure as defined in IEEE Std 260.1-2004 [B2].	Optional
unitQualifier	<i>c:NonBlankString</i>	A textual qualifier that is to be applied to the attribute of either the 'standardUnit' or the 'nonStandardUnit.' Examples include "rms" or "peak to peak" for a standard unit of "volts."	Optional

Attributes 'standardUnit' and 'nonStandardUnit' shall be alternatives. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

Attributes 'standardUnit', 'nonStandardUnit,' and 'unitQualifier' are defined in the attribute group *c:UnitAttributes*, which is defined in B.1 of IEEE Std 1671.

4.4.30.2 Child elements

Name	Type	Description	Use
Confidence	<i>xs:double</i>		Optional
ErrorLimits	<i>c:Limit</i>		Optional
ItemDescription	Type defined inline	See 4.4.31.	0..∞
Range	<i>c:Limit</i>		Optional
Resolution	<i>xs:double</i>		Optional

Elements *Resolution*, *ErrorLimits*, *Range*, and *Confidence* are defined in the group *c:DatumQuality*, which is defined in B.1 of IEEE Std 1671.

4.4.31 CollectionDescription/ItemDescription

Describes one data item.

Base type: extension of *ValueDescription*, see 4.4.310

4.4.31.1 Attributes

Name	Type	Description	Use
name	<i>c:NonBlankString</i>	Name of the data item.	Optional

4.4.31.2 Child elements

This element definition contains no child elements.

4.4.32 complexArrayDescription

Base type: extension of *IndexedArrayDescriptionType*, see 4.4.116

Describes an array of complex numeric values. Where applicable, can be assigned to elements of type *IndexedArrayDescriptionType*.

Note that it is possible to have elements with this type that have no elements or attributes. The presence of such an element indicates that the described entity is an array with unknown size and number of dimensions where elements have no unit, that a default value for elements is not specified, and that a nominal value for the described entity is not specified.

4.4.32.1 Attributes

This type definition contains no attributes.

4.4.32.2 Child elements

Name	Type	Description	Use
DefaultElementValue	<i>c:complex</i>	Default value, applies to all array elements whose value is not explicitly defined through <i>NominalElementValue</i> child elements.	Optional
NominalElementValue	Type defined inline	See 4.4.33.	0..∞

4.4.33 complexArrayDescription/NominalElementValue

The value of an array element, for the nominal array value.

Base type: extension of *c:complex*

4.4.33.1 Attributes

Name	Type	Description	Use
position	<i>c:ArrayIndexor</i>	Index of the element.	Required

4.4.33.2 Child elements

This element definition contains no child elements.

4.4.34 complexDescription

Base type: extension of *DatumDescriptionType*, see 4.4.48

Describes a complex numeric value (i.e., a value that has real and imaginary components). Where applicable, can be assigned to elements of type *DatumDescriptionType*.

4.4.34.1 Attributes

This type definition contains no child attributes.

4.4.34.2 Child elements

Name	Type	Description	Use
NominalValue	<i>c:complex</i>	Nominal value.	Optional

4.4.35 Component

Base type: extension of *c:ItemDescription*

Shall describe one component of the UUT.

4.4.35.1 Attributes

Name	Type	Description	Use
ID	<i>c:NonBlankString</i>	ID of the pin where the fault occurs. References a connector pin defined under the Component ancestor.	Required

4.4.35.2 Child elements

Name	Type	Description	Use
Interface	<i>c:PhysicalInterface</i>	Shall define the interface of the component. For components that are regular electronic parts, a single “dummy” <i>Connector</i> shall be defined, with minimal information attached. Additional information can be provided and multiple <i>Connector</i> elements can be used when describing “daughter card” components.	Optional

4.4.36 Component/Interface

Shall define the interface of the component. For components that are regular electronic parts, a single “dummy” *Connector* element shall be defined, with minimal information attached. Additional information can be provided, and multiple *Connector* elements can be used when describing “daughter card” components.

NOTE—This section describes the constraints associated with the element. The element’s type is described in Annex B of IEEE Std 1671.

4.4.36.1 Attributes

Refer to Annex B of IEEE Std 1671 for a description of the element’s attributes.

4.4.36.2 Child elements

Refer to Annex B of IEEE Std 1671 for a description of this element’s child elements.

4.4.36.3 Constraints

Name	Type	Refers	Selector	Fields
componentConnectorKey	<i>Key</i>		<i>/c:Connectors/c:Connector</i>	@ID
componentConnectorRef	<i>Keyref</i>	componentConnectorKey	<i>/c:Ports/c:Port/c:ConnectorPins/c:ConnectorPin</i>	@connectorID
componentPortNamesUnique	<i>Unique</i>	componentConnectorKey	<i>/c:Ports/c:Port</i>	@name

4.4.37 Condition

Shall describe a precondition or postcondition.

4.4.37.1 Attributes

Name	Type	Description	Use
Name	<i>c:NonBlankString</i>	Name of the precondition or postcondition.	Optional

4.4.37.2 Child elements

Name		Type	Description	Use
Choice	Other	<i>c:NonBlankString</i>	Shall describe the precondition or postcondition through free-form text.	Required
	StateVariableValue	Type defined inline	See 4.4.38.	
NOTE—Elements <i>StateVariableValue</i> and <i>Other</i> are alternative elements.				

4.4.38 Condition/StateVariableValue

Shall describe the precondition or postcondition in terms of state variable values.

4.4.38.1 Attributes

Name	Type	Description	Use
stateVariableValueID	<i>c:NonBlankString</i>	ID of the state variable value.	Required

4.4.38.2 Child elements

This element definition contains no child elements.

4.4.39 Conditions

Shall describe the preconditions and postconditions for an action (test or session action) or a test group. Refer to B.3 for an example where conditions are used. Refer to B.3.2 for usage information regarding conditions.

4.4.39.1 Attributes

This type definition contains no attributes.

4.4.39.2 Child elements

Name	Type	Description	Use
PostConditions	<i>PostConditions</i> , see 4.4.218	Shall describe all postconditions for the action or test group. Preconditions can be fulfilled by executing actions or test groups with the appropriate postconditions. For example, a session action can apply power, download firmware, or change the temperature of a thermal chamber; a test can verify the signature of the UUT, and so on.	Optional
PreConditions	<i>PreConditions</i> , see 4.4.221	Shall describes all preconditions for the action or test group. The specification of preconditions aims to ensure that actions and test groups are executed only if the proper conditions exist. For example, power is applied to the UUT, the appropriate firmware is loaded in the UUT, the UUT temperature is within specified limits, and so on.	Optional

4.4.40 Connection

Shall define a signal connection.

4.4.40.1 Attributes

This type definition contains no attributes.

4.4.40.2 Child elements

Name	Type	Description	Use
Networks	Type defined inline	See 4.4.41.	Required
Ports	Type defined inline	See 4.4.42.	Required

4.4.41 Connection/Networks

Shall describe the connection of ports defined under the parent element to ports that belong to interface connectors, test points, and components of the UUT.

4.4.41.1 Attributes

This element definition contains no attributes.

4.4.41.2 Child elements

Name	Type	Description	Use
Network	<i>hc:Network</i>	Shall describes one connection. One <i>Node</i> child shall reference a <i>Port</i> defined under the <i>Connection</i> ancestor. Other <i>Node</i> children can reference <i>Ports</i> of interface connectors, test points, or components.	1..∞

4.4.42 Connection/Ports

Shall define the ports that are connected. These are signal/logical ports (e.g., “HI,” “LO,” “A,” “B,” “C,” “N,” “TX,” “RX,” “GND,” etc.).

4.4.42.1 Attributes

This element definition contains no attributes.

4.4.42.2 Child elements

Name	Type	Description	Use
Port	<i>c:Port</i>	Shall define one port.	1..∞

4.4.43 ConnectionDatum

Base type: extension of *c:DatumType*

Shall define a signal connection. Where applicable, can be assigned to elements of type *c:DatumType*. The optional attributes and child elements of the base type are not applicable and shall not be used.

4.4.43.1 Attributes

This type definition contains no attributes.

4.4.43.2 Child elements

Name	Type	Description	Use
Connection	<i>Connection</i> , see 4.4.40	Shall define the connection.	Required

4.4.44 ConnectionDatumDescription

Base type: extension of *DatumDescriptionType*, see 4.4.48

Shall describe a signal connection, without defining the actual ports. Where applicable, can be assigned to elements of type *DatumDescriptionType*. The optional attributes and elements of the base type are not applicable and shall not be used.

4.4.44.1 Attributes

This type definition contains no attributes.

4.4.44.2 Child elements

Name	Type	Description	Use
NumberOfPorts	<i>xs:positiveInteger</i>	Shall indicate the number of ports. This optional element shall appear only if the described connection contains a fixed number of ports.	Optional

4.4.45 dateTimeArrayDescription

Base type: extension of *IndexedArrayDescriptionType*, see 4.4.116

Describes a date-time value. Where applicable, can be assigned to elements of type *IndexedArrayDescriptionType*.

Note that it is possible to have elements with this type that have no elements or attributes. The presence of such an element indicates that the described entity is an array with unknown size and number of dimensions where elements have no unit, that a default value for elements is not specified, and that a nominal value for the described entity is not specified.

4.4.45.1 Attributes

This type definition contains no attributes.

4.4.45.2 Child elements

Name	Type	Description	Use
DefaultElementValue	<i>c:dateTime</i>	Default value, applies to all array elements whose value is not explicitly defined through <i>NominalElementValue</i> child elements.	Optional
NominalElementValue	Type defined inline	See 4.4.46.	0..∞

4.4.46 dateTimeArrayDescription/NominalElementValue

Base type: extension of *c:dateTime*

The value of an array element, for the nominal array value.

4.4.46.1 Attributes

Name	Type	Description	Use
position	<i>c:ArrayIndexor</i>	Index of the element.	Required

4.4.47 dateTimeDescription

Base type: extension of *DatumDescriptionType*, see 4.4.48

Describes a date-time value. Where applicable, can be assigned to elements of type *DatumDescriptionType*.

4.4.47.1 Attributes

This type definition contains no attributes.

4.4.47.2 Child elements

Name	Type	Description	Use
NominalValue	<i>c:dateTime</i>	Nominal value.	Optional

4.4.48 DatumDescriptionType (abstract)

Abstract base type for types that describe a value consisting of a data item (e.g., numeric, boolean, string, and date-time), an optional nominal value, optional unit information, and optional quality information (range, resolution, and accuracy). The following derived types are defined in the schema:

- a) *binaryDescription*, see 4.4.26
- b) *booleanDescription*, see 4.4.29
- c) *complexDescription*, see 4.4.34
- d) *dateTimeDescription*, see 4.4.47
- e) *doubleDescription*, see 4.4.78
- f) *hexadecimalDescription*, see 4.4.95
- g) *integerDescription*, see 4.4.123
- h) *octalDescription*, see 4.4.134
- i) *stringDescription*, see 4.4.255
- j) *unsignedIntegerDescription*, see 4.4.298

Note that this is an abstract type, used as a base type for other type definitions. Where this type is assigned in the schema, elements with derived types shall be provided. The ‘xsi:type’ attribute shall be used to indicate the type of the elements.

Attributes ‘standardUnit’ and ‘nonStandardUnit’ shall be alternatives. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

4.4.48.1 Attributes

Name	Type	Description	Use
nonStandardUnit	<i>c:NonBlankString</i>	Nonstandard unit of the data item. This attribute supports any nonstandard unit, not already defined in IEEE Std 260.1-2004 [B2].	Optional
standardUnit	<i>c:StandardUnit</i>	Standard unit of the data item. When used, this attribute shall only contain a unit of measure as defined in IEEE Std 260.1-2004 [B2].	Optional
unitQualifier	<i>c:NonBlankString</i>	A textual qualifier that is to be applied to the attribute of either the ‘standardUnit’ or ‘nonStandardUnit.’ Examples include “rms” or “peak-to-peak” for a standard unit of “volts.”	Optional

Attributes ‘standardUnit’ and ‘nonStandardUnit’ shall be alternatives. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.

Attributes ‘standardUnit,’ ‘nonStandardUnit,’ and ‘unitQualifier’ are defined in the attribute group *c:UnitAttributes*, which is defined in B.1 of IEEE Std 1671.

4.4.48.2 Child elements

Name	Type	Description	Use
Confidence	<i>xs:double</i>		Optional
ErrorLimits	<i>c:Limit</i>		Optional
Range	<i>c:Limit</i>		Optional
Resolution	<i>xs:double</i>		Optional

Elements Resolution, ErrorLimits, Range, and Confidence are defined in the group *c:DatumQuality*, which is defined in B.1 of IEEE Std 1671.

4.4.49 DetailedConnectorLocation

Base type: extension of *c:ConnectorLocation*

Shall identify the location of a port on a specific connector pin, while providing detailed information for that pin.

The type shall be used when it is necessary to provide detailed information for the connector pin. Otherwise, the base type *c:ConnectorLocation* shall be used instead.

An element with this type shall not appear in instance documents unless it has at least one child element. Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically. See B.2 for an example of using this complex type.

4.4.49.1 Attributes

This type definition contains no attributes.

4.4.49.2 Child elements

Name	Type	Description	Use
DataAccess	<i>c:NonBlankString</i>	Shall describe “virtual pins,” representing firmware variables that may be read during testing (similar to measurements performed on physical pins). Shall specify the command used to retrieve data from the firmware variable.	Optional
Extension	<i>c:Extension</i>	Provided to permit schema extensibility.	Optional
Function	<i>PinFunction</i> , see 4.4.217	Shall describe the function of the pin, including pin designation, signal name, signal definition and signal description. This element shall not appear for pins of type “Not Connected.” Note that this constraint is not enforced by the XML schema. Consumers should verify the constraint programmatically.	Optional
Wire	Type defined inline	See 4.4.50.	Optional

4.4.50 DetailedConnectorLocation/Wire

Shall contain information for the wire that will be connected to the UUT.

4.4.50.1 Attributes

This element definition contains no attributes.

4.4.50.2 Child elements

Name	Type	Description	Use
Description	<i>c:NonBlankString</i>	Shall contain a free-form description of the wire, including wire type (e.g., coaxial), wire size, and so on.	Required
Relationships	Type defined inline	See 4.4.51.	Optional

4.4.51 DetailedConnectorLocation/Wire/Relationships

Shall describe wire dependencies (e.g., “twisted with” or “shielded with”).

4.4.51.1 Attributes

This element definition contains no attributes.

4.4.51.2 Child elements

Name	Type	Description	Use
Relationship	Type defined inline	See 4.4.52.	1..∞

4.4.52 DetailedConnectorLocation/Wire/Relationships/Relationship

Shall describe one wire dependency.

4.4.52.1 Attributes

This element definition contains no attributes.

4.4.52.2 Child elements

Name	Type	Description	Use
Description	<i>c:NonBlankString</i>	Shall contain a free-form description of the relationship (e.g., “twisted with” or “shielded with”).	Required
Wires	<i>hc:Network</i>	Shall identify the other wire or wires the current wire is in relationship with by referencing one or more ports of the UUT interface.	Required

4.4.53 DetailedTestInformation

Shall describe initialization, performance, and diagnostic test information.

Figure 2 illustrates the top-level XML complex types that comprise the *DetailedTestInformation* type and the other IEEE standards these types reference. The referenced subclause identifies where the definition of the element is located within this standard.

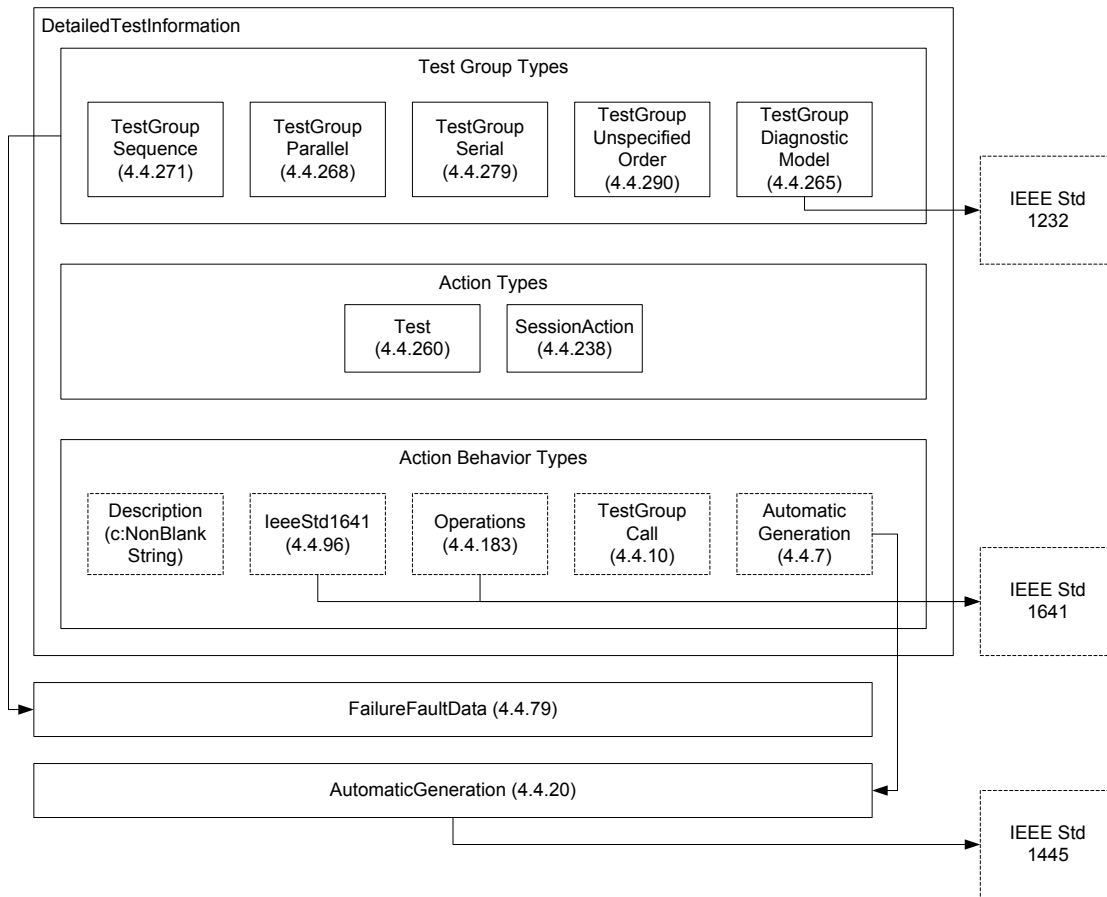


Figure 2—DetailedTestInformation type

4.4.53.1 Attributes

This type definition contains no attributes.

4.4.53.2 Child elements

Name	Type	Description	Use
Actions	Type defined inline	See 4.4.54.	Required
EntryPoints	Type defined inline	See 4.4.56.	Required
Extension	<i>c:Extension</i>	Provided to permit schema extensibility.	Optional
GlobalSignals	Type defined inline	See 4.4.61.	Optional
StateVariables	Type defined inline	See 4.4.64.	Optional
TestGroups	Type defined inline	See 4.4.69.	Required

4.4.54 DetailedTestInformation/Actions

Shall contain the list of actions (tests and session actions) described by the instance document.

4.4.54.1 Attributes

This element definition contains no attributes.

4.4.54.2 Child elements

Name	Type	Description	Use
Action	<i>ActionType</i>	Shall describe one action (test or session action).	1..∞

4.4.54.3 Constraints

Name	Type	Refers	Selector	Fields
actionNamesUnique	<i>Unique</i>		/Action	@name

4.4.55 DetailedTestInformation/Actions/Action

Shall describe one action (test or session action). Note that *ActionType* is an abstract type. Consequently, elements with derived types shall be provided. The ‘xsi:type’ attribute shall be used to indicate the type of these elements. The following derived types are defined in the schema:

- a) *Test*, see 4.4.260
- b) *SessionAction*, see 4.4.238

NOTE—This clause describes the constraints associated with the element. Refer to 4.4.5 for a description of the element’s type.

4.4.55.1 Attributes

Refer to 4.4.5.1 for a description of the element’s attributes.

4.4.55.2 Child elements

Refer to 4.4.5.2 for a description of this element’s child elements.