



IEEE Recommended Practice for Defining a Commonly Acceptable Musical Application Using XML

IEEE Computer Society

Sponsored by the
Standards Activities Board

1599TM

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
26 September 2008

IEEE Std 1599TM-2008

IEEE Recommended Practice for Defining a Commonly Acceptable Musical Application Using XML

Sponsor

**Standards Activities Board
of the
IEEE Computer Society**

Approved 12 June 2008

IEEE-SA Standards Board

Abstract: This recommended practice develops an XML application defining a standard language for symbolic music representation. The language is a meta-representation of music information for describing and processing said music information within a multi-layered environment, for achieving integration among structural, score, Musical Instrument Digital Interface (MIDI), and digital sound levels of representation. Furthermore, the recommended practice integrates music representation with already defined and accepted common standards. This recommended practice will be accepted by any kind of software dealing with music information, e.g., score editing, optical music recognition (OMR) systems, music performance, musical databases, and composition and musicological applications.

Keywords: description, multimedia, music, synchronization, XML

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

Copyright © 2008 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 26 September 2008. Printed in the United States of America.

IEEE and POSIX are registered trademarks in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

Windows and MS-DOS are registered trademarks of Microsoft Corporation in the United States and other countries.

PDF: ISBN 978-0-7381-5774-0 STD95809
Print: ISBN 978-0-7381-5775-7 STDPD95809

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.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The 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 the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the 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. The 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 this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, 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 herein 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 five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, 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 this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon his or her 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.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received 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 interpretation requests except in those cases where the matter has previously received formal consideration. 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 interpretation of the 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, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be submitted to the following address:

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

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 1599-2008, IEEE Recommended Practice for Defining a Commonly Acceptable Musical Application Using XML.

This introduction provides some background on the rationale used to develop this recommended practice. It is meant to aid in the understanding and usage of this document.

This recommended practice can be considered a first collective step toward the improvement of both sub-symbolic and symbolic music coding and processing, music communication, and capabilities of individuals and companies in the following areas:

- Music processing and delivering, at multiple levels of representation
- Modeling for music analysis and synthesis
- Definition and implementations of tools for automatic music understanding, for composing and for performing music, for music browsing on the Web

Topics of major interest for the definition of this proposal are as follows:

- Realization of tools, test, demonstrations
- Support of existing applications and formats
- Semantics of music notation, compatibility with Common Western Notation (CWN)
- Graphical notation, representation, interleaving for real-time interaction
- Orientation to networks
- Integration of symbolic notation, audio, visual, metadata, representation
- Possibility of splitting: fragmentation, segmentation
- Orientation to musicology and/or performances

Brief history of this project and relationship to the Intelligent Manufacturing Systems financial support

The Project Authorization Request (PAR) to define a new recommended practice was submitted to the IEEE Computer Society (CS) Standards Activity Board (SAB) and accepted in September 2001.

The abstract was submitted to the Intelligent Manufacturing Systems (IMS) secretariats and accepted in August 2002 (second version). IMS is a global fund consisting of seven regions: U.S., Canada, Australia, Japan, Korea, Switzerland, and the European Union. A project must have partners from at least three different regions and must be accepted by all secretariats.^a

The IEEE Conference *MAX 2002: Musical Applications Using XML* took place in Milan, Italy, on September 19–20, 2002, organized by the IEEE CS Working Group of P1599.^b During development, MX was the internal code for the project that became IEEE Std 1599 (see 1.5).

Various meetings of the IEEE CS SAB were attended from 2002 through 2006. Slides were presented regarding the progress of the project at every meeting. Denis Baggi, Working Group Chair and one of the authors of this recommended practice, became a member of the SAB, functioning as liaison to the

^a See the IMS Web site: <http://www.ims.org>.

^b The P before 1599 indicates an IEEE authorized standards project that was not approved by the IEEE-SA Standards Board at the time the Working Group was formed.

Technical Activity Board (TAB), of which he was already a member. The IEEE CS put an additional Web page for the project on its site, which was linked to the IMS Web page.

The proposal for a feasibility study, “Music XML Lite,” was accepted by the Swiss IMS Secretariat, and the project ran from February through November of 2004. This included a trip to the U.S., Australia, and Japan to meet potential project partners. An intermediate report was delivered in September 2004 that outlined the past, present, and future of the project. Various talks were given and leaflets distributed about the project, e.g., at the IMS Forum in Cernobbio, Italy, on May 14, 2004. In particular, two talks were given at an IMS meeting in Gold Coast, Australia, on November 9, 2004.

IMS endorsed the project in 2006, and the Commission for Technological Innovation of the Department of Public Economy of the Swiss Federal Government accepted a proposal for financing in 2006. The project consisted of completion of the standard definition as well as realizing applications for the *Radio Televisione della Svizzera Italiana*^c and for the jazz festival *Jazz Ascona–New Orleans and Classics*,^d including its sponsored programs to help the victims of the Katrina hurricane.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard 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.

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 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 Standards Association Web site at <http://ieeexplore.ieee.org/xpl/standards.jsp>, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA Web site at <http://standards.ieee.org>.

^c See the RTSI Web site: <http://www.rtsi.ch>.

^d See <http://www.jazzascona.com/>.

Errata

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

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

Patents

Attention is called to the possibility that implementation of this recommended practice may require use of subject matter covered by patent rights. By publication of this recommended practice, no position is taken with respect to the existence or validity of any patent rights in connection therewith. 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 recommended practice 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 recommended practice was submitted to the IEEE-SA Standards Board for approval, the IEEE P1599 Working Group had the following membership:

Denis L. Baggi, *Chair*
Goffredo Haus, *Official Reporter*

Antonello D'Aguanno
Adriano Baratè
Lazlo Belady
Michael Cohen
Loretta Diana
Federico Flueckiger

Tosiyasu L. Kunii
Maurizio Longari
Luca A. Ludovico
Damian McGuckin
Tony Milosz
Kalpdrum Passi
Alberto Pinto

Perry Roland
Elisa Russo
Donald Sloan
Lorenzo Sommaruga
Paolo Tagliolato
Giancarlo Vercellesi

Working Group members who contributed in the past include the following:

Liz Armstrong
Jérôme Barthélémy
France Champagne
Gustavo Frederico
Michael Good

Charl Jeannot
Markus Lepper
Yashuo Mishiro
Marco Mattavelli
Dan Nagy

Jacques Steyn
Tony Strasser
Guy Tremblay
Ruo-hua Zhou
Giorgio Zoia

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

Juan Carreon
James Case
Clint Chaplin
Keith Chow
John Cole
Tommy Cooper
Thomas Dineen
Paul Eastman
Thomas Gauntner

Michael Geipel
Randall Groves
Werner Hoelzl
Mark Jaeger
Piotr Karocki
Mark Knight
Susan Land
William Lumpkins
Gary Michel
Michael S. Newman

Charles Ngethe
Jose Puthenkulam
Bartien Sayogo
Richard Schrenker
David J. Schultz
John Sheppard
John Vergis
P. Wolfgang
Oren Yuen

When the IEEE-SA Standards Board approved this recommended practice on 12 June 2008, it had the following membership:

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

Victor Berman
Richard DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
William R. Goldbach
Arnold M. Greenspan
Kenneth S. Hanus

Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Glenn Parsons
Ronald C. Petersen

Chuck Powers
Narayanan Ramachandran
Jon Walter Rosdahl
Anne-Marie Sahazizian
Malcolm V. Thaden
Howard L. Wolfman
Don Wright

*Member Emeritus

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

Satish K. Aggarwal, *NRC Representative*
Michael H. Kelley, *NIST Representative*

Malia Zaman
IEEE Standards Program Manager, Document Development

Lorraine Patsco
IEEE Standards Program Manager, Technical Program Development

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Benefits	2
1.4 Official Web site	2
1.5 Registration object	2
2. Normative references	2
3. Acronyms and abbreviations	2
4. Conventions	3
5. Representation of music information through XML	3
6. IEEE 1599 layers	4
6.1 General	6
6.2 Logic	11
6.3 Structural	73
6.4 Notational	79
6.5 Performance	82
6.6 Audio	89
Annex A (informative) Bibliography	93
Annex B (informative) IEEE 1599 DTD	94
Annex C (informative) Running applications to demonstrate IEEE 1599	110

IEEE Recommended Practice for Defining a Commonly Acceptable Musical Application Using XML

IMPORTANT NOTICE: *This standard is not intended to assure safety, security, health, or environmental protection in all circumstances. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.*

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 Scope

This recommended practice will develop an XML application defining a standard language for symbolic music representation. The language will be a meta-representation of music information for describing and processing said music information within a multi-layered environment, for achieving integration among structural, score, Musical Instrument Digital Interface (MIDI), and digital sound levels of representation. Furthermore, the recommended practice should integrate music representation with already defined and accepted common standards. The recommended practice will be accepted by any kind of software dealing with music information, e.g., score editing, optical music recognition (OMR) systems, music performance, musical databases, and composition and musicological applications.

1.2 Purpose

There is currently no defined, independent standard for representing music information that can describe and process all the different layers that characterize music information. For each layer of music information, there is one or more accepted standard [e.g., MIDI for performances, Notation Interchange File Format (NIFF) for notation, and so on] and/or one or more proprietary formats. None of these can be suitably applied to other layers. This recommended practice will make easier the integration, interchange, and translation from one layer to another of music information across different applications and even for different users.

1.3 Benefits

There are many advantages in this recommended practice. Musicians could benefit because they would be able to handle each representation of a music piece as a unique entity—for example, by simultaneously watching a score and listening to the music of a particular audio recording. Music analysts would have the complete control over notational information, for example, by easily comparing the notational content with the aural features of a recording. Music vendors could access a database where each piece of music would be stored along with different instances of its rendition, their own different properties related to a single product. Music producers would benefit from this work because they could set up a music database in which all related material could be handled in a structured way. For example, a Web service could be created to which users connect and select a piece of music, accessing a MIDI rendering of it for practicing, along with and a visualization of the score. The integration of all music aspects would also improve applications of human-machine interaction, thus expanding the possibilities in the areas of digital music composition and realization.

1.4 Official Web site

IEEE 1599 is an XML-based format and is the code for this recommended practice. The Document Type Definition (DTD) of the current release is available from the IEEE Web site.¹

The official Laboratory of Musical Informatics (LIM) Web site² contains the latest news.

1.5 Registration object

MX was the object registered by the LIM of the State University of Milan, Italy. Reference to it implies a reference to the contents of this recommended practice.

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 referenced, the latest edition of the referenced document (including any amendments or corrigenda) applies. This recommended practice shall be used *in conjunction with* the following publications:

Extensible Markup Language (XML) 1.0, Fourth Edition (August 2006).³

Scalable Vector Graphics (SVG) 1.1 Specification (January 2003).⁴

XML Schema 1.1 (May 2001).⁵

3. Acronyms and abbreviations

CWN Common Western Notation

DTD Document Type Definition

¹ Available at <http://standards.ieee.org/downloads/1599/1599-2008/>.

² See <http://www.mx.dico.unimi.it>.

³ Available from the W3C at <http://www.w3.org/TR/xml>.

⁴ Available from the W3C at <http://www.w3.org/TR/SVG11>.

⁵ Available from the W3C at <http://www.w3.org/XML/Schema>.

LIM	Laboratorio di Informatica Musicale (Laboratory of Music Informatics)—Dipartimento di Informatica e Comunicazione (Informatics and Communication Dept.)—Università degli Studi di Milano
LOS	Logically Organized Symbols
MIDI	Musical Instrument Digital Interface
MX	internal code previously used for ieee1599
PNML	Petri Net Markup Language
SMDL	Standard Music Description Language
SMI	Symbolic Music Information, generic term for symbolic representation of music
VTU	virtual time unit
XML	Extensible Markup Language

4. Conventions

In the following, an XML-based language will be defined as the result of this recommended practice. This language is named IEEE 1599, representing the music-oriented application of XML.

In the DTD that defines the language, the following naming conventions have been used:

- The names of elements and attributes are always in small letters, even when they contain acronyms such as MIDI.
- When composite names are used, their parts are separated by the _ (underscore) character.

In this document, the following typographic conventions will be employed:

- The name of layers, when cited inside the text, starts with a capital letter, is highlighted in italic font, and does not require an article (e.g., *General* layer).
- Source code and XML is presented using a typewriter-like font (e.g., `IEEE 1599`).
- Text contents of elements and values of attributes are written in standard character, among double quotes (e.g., “value”).
- Extracts and examples from DTD are embedded inside text boxes surrounded by a thin black border.
- Images are surrounded by a thin black border.

5. Representation of music information through XML

This recommended practice is based on six *layers* into which music information can be organized: *General*, *Logic*, *Structural*, *Notational*, *Performance*, and *Audio*. These aspects are called “layers” because each represents a different level of representation and abstraction of music information. However, they can be viewed as a single entity in the context of Symbolic Music Information (SMI). SMI relates all existing representations such as notational, performance, and audio layers using the logical and structural layers of music.

There exist file formats to describe single aspects of music information, but there is no format able to represent all heterogeneous contents. Since neither preexisting formats for music encoding can be ignored

nor is it possible to provide a description for all aspects in one single format, a meta-language such as XML has been employed. This provides a description of all related music elements and links the corresponding media objects already encoded. Thus, music contents are either encoded in the proper layers (*General*, *Logic*, and *Structural* layer actually store information) or they remain in their original format, with links from the corresponding layers to files (*Notational*, *Performance*, and *Audio* layers store mappings toward external files). Media files are handled as they are, and media contents are expressed in their original encoding. The comprehensive format described in this recommended practice uses layers to represent the relationships between music events and their occurrence in media files, thus allowing an overall synchronization.

The key characteristics of such language can be summarized as follows:

- Richness in the multimedia descriptions related to the same music piece (symbolic, textual, graphical, audio, and video contents);
- Possibility to describe or link different media objects of the same type—for instance, many performances of the same piece or many score scans from different editions.

In order to organize heterogeneous contents, a multi-layer structure has been employed, in which each layer corresponds to a different kind of information.

In music, there is an intrinsic space-time relationship, which is not explicit in many existing representations, that can be seen as a bi-directional mapping function between space and time. An underlying structure called *spine*, which acts as a backbone for a music, holds all layers together like glue. It is composed of *events*, each of which may have a reference in the time domain and one in the space domain. Using this concept of spine, a method can be defined that joins each different file format into a single structure.

In the following subclauses, an XML DTD will be briefly surveyed. This format has been designed for the representation of music information. Then, the concepts of layered SMI and its representation in XML syntax will be described.

6. IEEE 1599 layers

This clause describes each of the layers of IEEE 1599. Every element and attribute is described hierarchically. The conceptual hierarchy of layers is shown in Figure 1. Annex B shows the complete DTD of IEEE 1599.

Here follow the *entities* commonly employed in IEEE 1599.

```
<!ENTITY % svg
  PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
%svg;

<!ENTITY % ChannelRequired "#REQUIRED">
<!ENTITY % mididtd
  PUBLIC "-//MIDI Manufacturers Association//DTD MIDIEvents 1.0//EN"
  "http://www.midi.org/dtds/MIDIEvents10.dtd" >
%mididtd;

<!ENTITY % spine_ref
  "event_ref IDREF #REQUIRED">

<!ENTITY % spine_start_end_ref
```

```
"start_event_ref IDREF #REQUIRED
end_event_ref IDREF #REQUIRED">

<!ENTITY % accidental
    "(none | double_flat | flat | natural | sharp |
      double_sharpdouble_flat | flat_and_a_half | flat | demiflat |
      natural | demisharp | sharp | sharp_and_a_half | double_sharp)">

<!ENTITY % formats
    "(application_excel | application_mac-binhex40 |
      application_msword | application_octet-stream | application_pdf |
      application_x-director | application_x-gzip |
      application_x-javascript | application_x-macbinary |
      application_x-pn-realaudio | application_x-shockwave_flash |
      application_x-tar | application_zip | audio_aiff | audio_avi |
      audio_mp3 | audio_mpeg | audio_mpeg3 | audio_mpg | audio_wav |
      audio_x_aiff | audio_x_midi | audio_x_wav | audio_x-mp3 |
      audio_x-mpeg | audio_x-mpeg3 | audio_x-mpegaudio | audio_x-mpg |
      audio_x-ms-wma | image_avi | image_bmp | image_x-bmp | image_x-bitmap |
      image_x-xbitmap | image_x-win-bitmap | image_x-windows-bmp |
      image_ms-bmp | image_x-ms-bmp | application_bmp | application_x-bmp |
      application_x-win-bitmap | application_preview | image_gif |
      image_jpeg | image_pict | image_png | application_png |
      application_x-png | image_tiff | text_html |
      text_plain_application_postscript | video_avi | video_mpeg |
      video_msvideo | video_quicktime | video_x-msvideo | video_x-ms-wmv |
      video_x-qtz | video_xmpg2)">
```

The purpose and meaning of the entities is explained further as follows in the context of attributes and values.

Entities

- `spine_ref`, standing for a required reference to IDs specified in spine sub-layer.
- `spine_start_end_ref`, standing for two required references to IDs specified in the spine sub-layer. Usually this entity is used when a sign has a temporal or spatial development, and it requires a reference both to a start and to an end event.
- `accidental`, listing all the supported accidentals. Please note that microtonal music is supported too, thanks to the accidentals that lower/raise the pitch by one or many quarter tones.
- `formats`, listing all the supported file formats and encoding formats in a way that resembles MIME types.

```
<!ELEMENT ieee1599 (general, logic, structural?, notational?,
performance?,
                                                                    audio?)>

<!ATTLIST ieee1599
    version CDATA #REQUIRED
    creator CDATA #IMPLIED>
```

Element `ieee1599` represents the root element of any IEEE 1599 file. Since XML organizes information in a hierarchical structure, each layer can be represented as a sub-tree of the root element.

Note that, among supported layers, only *General* and *Logic* are required.

Attributes

- `version`, the DTD version employed to create the file. At this stage, RC1 (standing for Release Candidate 1) should be its default value.
- `creator`, the author (software or human user) of the IEEE 1599 encoding.

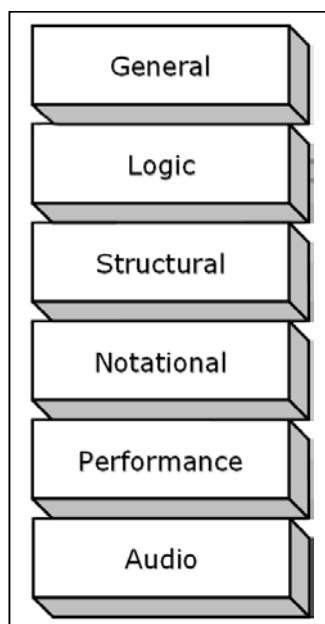


Figure 1—IEEE 1599 layers

6.1 General

```
<!ELEMENT general (description, related_files?, analog_media?, notes?,  
rights?)>
```

General layer provides a general description of the music work and groups information on all the related instances. General information regarding the music work as a whole should be stored in this layer. General information about particular instances (e.g., the performers of an audio track) is treated in the corresponding layer.

6.1.1 Description

```
<!ELEMENT description (main_title, author*, other_title*, number?,  
work_title?, work_number?, date*, genres?)>
```

Element `description` contains catalog metadata about the piece.

6.1.1.1 Main_title

```
<!ELEMENT main_title (#PCDATA)>
```

Element `main_title` contains the title of the piece. A main title is required for IEEE 1599 validity. When the title is not known, the content can be empty or it can conventionally contain the string

“unknown.” IEEE 1599 is meant to represent single pieces or movements, so contents such as “Lovely Rita” (rather than “Sgt. Pepper’s Lonely Hearts Club Band”) or “III movement—Allegro” (rather than “Piano sonata nr. 17”) are to be expected. The parenthesized contents will be the subject of another element, namely `work_title`.

6.1.1.2 Author

```
<!ELEMENT author (#PCDATA)>
<!ATTLIST author
  type CDATA #IMPLIED>
```

Element `author` can be used to list the name(s) of the author(s) of the piece.

Attributes

—`type`, the role of the author within the composition. Multiple values (such as “voice and guitar” or “composer and librettist”) are allowed, even if the corresponding atomized version (i.e., the use of a number of elements with the same content and different values of the attribute) is preferable.

6.1.1.3 Other_title

```
<!ELEMENT other_title (#PCDATA)>
```

Element `other_title` can be used to represent alternative titles of the piece, such as translations of the title in other languages or traditional and popular names attributed to music pieces whose original title is different. For example, “Nel blu dipinto di blu” by Domenico Modugno is the original title (`main_title`) of a famous Italian song known worldwide as “Volare” (`other_title`); similarly, the “Air from Suite No. 3” (`main_title`) by Johann Sebastian Bach is usually known as “Air on the G string” (`other_title`).

6.1.1.4 Number

```
<!ELEMENT number (#PCDATA)>
```

Element `number` provides the position of the piece within the sorted collection of pieces it belongs to. For instance, “Lovely Rita” can be numbered 3B (i.e., song number 3 of the B-side) as in the track listing of the original album, or 10 as in the track listing on the CD. Similarly, “III movement—Allegro” can be numbered 3 within the collection represented by all the movements that compose a sonata.

As IEEE 1599 encodes single music pieces, element `number` together with the contents of `work_title` and `work_number` (see 6.1.1.5 and 6.1.1.6) can help reconstruct a complete music work made of many movements.

6.1.1.5 Work_title

```
<!ELEMENT work_title (#PCDATA)>
```

Element `work_title` contains the title of the collection the piece belongs to. It can be used to describe the album title for pop music as well as the title of the whole composition for “cultured” music. For instance, Carl Orff’s “Carmina Burana” represents the work title, whereas “O Fortuna” is the main title of the piece encoded in IEEE 1599.

6.1.1.6 Work_number

```
<!ELEMENT work_number (#PCDATA)>
```

Element `work_number` contains the catalog number of the whole composition this piece belongs to. For instance, if the IEEE 1599 file encodes the 2nd movement from Emperor Quartet by Joseph Haydn, `work_number`'s content is "op. 76 no. 3" (the catalog number of the whole quartet). Similarly, for Wolfgang Amadeus Mozart's "Eine kleine Nachtmusik," `work_number`'s content is "K 525."

6.1.1.7 Date

```
<!ELEMENT date (#PCDATA)>  
<!ATTLIST date  
  type CDATA #IMPLIED>
```

Element `date` is devoted to the music piece dating. Such date can correspond to the composition date, the publishing, the first public performance, etc. This detailed information is provided by the attribute `type`.

Attributes

— `type`, the type of dating according to the aforementioned meaning.

6.1.1.8 Genres and genre

```
<!ELEMENT genres (genre+)>  
  
<!ELEMENT genre EMPTY>  
<!ATTLIST genre  
  name CDATA #REQUIRED  
  description CDATA #IMPLIED  
  weight CDATA #IMPLIED>
```

Element `genres` is a container for one or more specifications of music genre. Element `genre` allows specification a music genre for classification purpose.

Attributes

- `name`, the term or locution that textually identifies the genre. A number of dimensions may be needed for classification purpose: historical period (e.g., "20th century"), critical classification (e.g., "Baroque"), type of composition (e.g., "Counterpoint"), geographical area (e.g., "Balkan music"), ensemble (e.g., "quartet"), etc.
- `description` provides an extended text description of the genre.
- `weight` defines through a numerical value the weight of this particular classification with respect to other classifications listed within element `genres`. For instance, a piece can be considered 0.7 classical and 0.3 romantic, such as Beethoven's late compositions. Note that a reference scale is not provided, in the sense that the sum of values for different `genre` elements do not have to reach a predefined value.

6.1.2 Related_files and related_file

```
<!ELEMENT related_files (related_file+)>  
  
<!ELEMENT related_file EMPTY>
```

```
<!ATTLIST related_file
  file_name CDATA #REQUIRED
  file_format %formats; #REQUIRED
  encoding_format %formats; #REQUIRED
  spine_start_ref IDREF #IMPLIED
  spine_end_ref IDREF #IMPLIED
  description CDATA #IMPLIED
  copyright CDATA #IMPLIED
  notes CDATA #IMPLIED>
```

Element `related_files` is a container for one or more specifications of `related_file`. Element `related_file` is devoted to linking an external digital object somehow related to the encoded piece but not directly related to the occurrence of music events. In other words, the objects linked within `related_files` element should not contain contents semantically related to the piece. This definition includes scans of photos, fashion plates, sketches, covers, costumes, etc.; this definition does not include score scans, music performances, audio tracks, etc., which are described in other layers.

Attributes

- `file_name`, the path to locate the digital object. The path should use only slash and backslash characters as directory separators. The value of this attribute should respect the syntax rules supported by POSIX[®] and MS-DOS[®]/Windows[®] 6 operating systems.
- `file_format`, extracted from a complete list of supported formats presented in the IEEE 1599 DTD. IEEE 1599 formats are similar to the corresponding MIME media type, but media type name (e.g., “image”) and subtype name (e.g., “jpeg”) are separated by an _ (underscore character). As a consequence, a MIME type such as “image/jpeg” is expressed in IEEE 1599 as “image_jpeg.”
- `encoding_format` (see `file_format` for allowed values). Usually there is no difference between file and encoding format, but some formats (such as TIFF for graphics and WAV for audio) can be seen as containers of a number of other formats. In this case, `file_format` reflects the “external” encoding, whereas `encoding_format` specifies the actual encoding within the container.
- `spine_start_ref` and `spine_end_ref` contain the identifiers of music events listed in the spine (see 6.2.1). These attributes allow synchronization of, respectively, the appearance and disappearance of digital objects with the occurrence of music events, and they are useful e.g., for slide show and multimedia presentations purposes.
- `description` provides an extended text description of the related file.
- `copyright` contains possible copyright information.
- `notes` provides an extra text field for detailed description and notes.

6.1.3 Analog_media and analog_medium

```
<!ELEMENT analog_media (analog_medium+)>

<!ELEMENT analog_medium EMPTY>
<!ATTLIST analog_medium
  description CDATA #REQUIRED
  copyright CDATA #IMPLIED
  notes CDATA #IMPLIED>
```

⁶ MS-DOS and Windows are registered trademarks of Microsoft Corporation in the United States and other countries. This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products may be used if they can be shown to lead to the same results.

Element `analog_media` is a container for one or more specifications of `analog_medium`. Element `analog_medium` is devoted to describe a non-digitized external digital object somehow related to the encoded piece. This definition includes costumes (intended as physical objects), hard copies of images and covers, and the like.

Attributes

- `description` provides an extended text description of the related medium.
- `copyright` contains possible copyright information.
- `notes` provides an extra text field for detailed description and notes.

6.1.4 Notes

```
<!ELEMENT notes (#PCDATA)>
```

Element `notes` can contain general notes about the piece, additional catalog information, software applications and hardware devices used in the coding process, and so on.

6.1.5 Complete examples

6.1.5.1 J. S. Bach—Inventio 1 from Zweistimmigen Inventionen

```
<general>
  <description>
    <main_title>Inventio 1</main_title>
    <author type="composer">Johann Sebastian Bach</author>
    <number>BWV 772</number>
    <work_title>Zweistimmigen Inventionen</work_title>
    <work_number>BWV 772-786</work_number>
    <genres>
      <genre name="Baroque" weight="100"/>
    </genres>
  </description>
</general>
```

6.1.5.2 A. Vivaldi—Largo from Concerto No. 1 in E major, Op. 8, RV 269

```
<general>
  <description>
    <main_title>Largo</main_title>
    <author type="composer">Antonio Vivaldi</author>
    <other_title>2nd movement</other_title>
    <number>2</number>
    <work_title>Concerto No. 1 in E major</work_title>
    <work_number>Op. 8, RV 269</work_number>
    <date>1723</date>
    <genres>
      <genre name="Baroque" weight="100"/>
    </genres>
  </description>
</general>
```

6.1.5.3 G. Puccini—Aria “E lucevan le stelle” from *Tosca*

```
<general>
  <description>
    <main_title>E lucevan le stelle</main_title>
    <author type="composer">Giacomo Puccini</author>
    <author type="librettist">Giuseppe Giacosa</author>
    <author type="librettist">Luigi Illica</author>
    <work_title>Tosca</work_title>
    <date type="first performance">14/1/1900 </date>
  </description>
</general>
```

6.2 Logic

```
<!ELEMENT logic (spine, los?, layout?)>
```

Logic layer provides music description from a symbolic point of view.

Logic layer represents the core of the IEEE 1599 format: it contains 1) the main time-space construct for localization and synchronization of music events, 2) the description of the score from a symbolic standpoint, and 3) information about a generic graphical implementation of symbolic contents. As a consequence, the inner structure of *logic* element is made of three sub-elements, namely *spine*, *los* (standing for Logically Organized Symbols), and finally *layout*. Note that only *spine* is strictly required, so that also IEEE 1599 supports music without symbolic contents (according to the standard CWN definition).

6.2.1 Spine

```
<!ELEMENT spine (event)+>
```

Spine is made of a sorted list of music events, where the definition and granularity of the term “event” can be defined by the author of the encoding. In this sense, *spine* represents an abstraction level, as the events identified do not have to correspond necessarily to score symbols, or audio samples, or anything else. The author can decide each time what goes under the definition of “music event,” according to the needs.

Spine represents a container for a sorted list of identifiers, expressed through a number of *event* elements (see 6.2.1.1).

The two-dimensional array of events is arranged as a one-dimensional ordered list. Even though notes, rests, and other symbols in a score are disposed both according to their vertical juxtaposition (related to harmony) and to their horizontal layout (corresponding to temporal evolution of music), the *spine* is a one-dimensional list. Referring each *spine* event to the previous one thanks to a path that goes through the music score as follows solves this difference. First, a vertical scan is performed, from upper symbols to lower symbols in a staff, and from upper staves to lower staves, and when all the simultaneous (i.e., vertically aligned) events have been considered, the next event on the right is considered and the vertical scan is resumed. Hence, the resulting list of events meanders through the score, from top to bottom and from left to right. Vertical alignments and horizontal offsets are defined by *hpos*, an attribute of *event* sub-element, as explained in 6.2.1.1.

IEEE 1599 is aimed not only at CWN, thus the problem of ordering music events in a one-dimensional sorted list exists also for non-traditional scores. In this case, a similar approach can be employed. For

instance, the occurrence of events in time can be managed by considering time offsets among non-synchronous events (non-zero value for `timing`) and simultaneity among synchronous events (`timing` = "0"). Finally, the time dimension originates the one-dimensional list.

As it will be shown in more detail, spine acts also as glue in a *multi-layer structure*, in which music events are not only listed and ordered by spine, but also tagged by unique identifiers. These identifiers, conceptually similar to unique key constraints in a database, are referred to by all instances of the corresponding event representations in other layers. In other words, spine references, and is referenced by, all other layers. This concept is illustrated in Figure 2, and constitutes the IEEE 1599 approach to global synchronization for heterogeneous media contents.

The presence of element `spine` for the IEEE 1599 file as well as the presence of at least one sub-element `event` are required for the validity of the encoding.

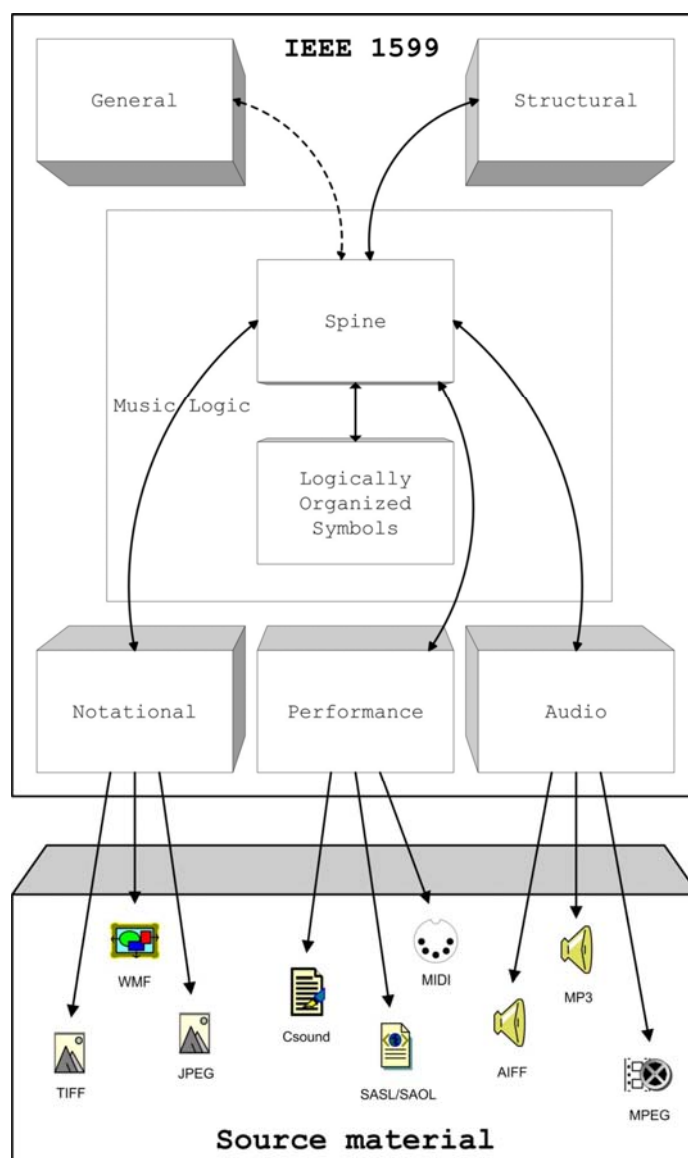


Figure 2—Relationships among IEEE 1599 layers

6.2.1.1 Event

```
<!ELEMENT event EMPTY>
<!--ATTLIST event
  id ID #REQUIRED
  timing CDATA "null"
  hpos CDATA "null">
```

Element `event` is one of the most important ones of the whole DTD. It defines and identifies an information unit that is referred to by other parts of the encoding. In a normal case, all chords and rests can be considered events and marked by an *id* through element `event`. However, the concept of music event is flexible.

First, even with standard CWN, not all events have to be described through spine events; hence, it is entirely possible to list the notes belonging to a part/voice only, or the quarter-notes only, or the tenor's super acute notes.

Second, aggregations of "atomic" music events can also be considered music events in some cases: for instance, in jazz improvisation usually only the harmonic grid is known, so the occurrence of a new chord could be considered the most meaningful music event, rather than the occurrence of single notes. In general, chords, measures, phrases, etc., can be defined as music events (and marked through an aggregate *id*) as necessary.

Finally, CWN is not required at all for IEEE 1599 validity, so that IEEE 1599 supports also non-traditional scores. In this case, the description provided in graphical or structural form can be linked to spine, no matter what definition is given to the concept of music event.

Attributes

- `id` provides a unique identifier within an IEEE 1599 file. Even if there is no convention about `id` naming in IEEE 1599, because of its importance in IEEE 1599 encoding, the value of this attribute should reflect music semantics. For instance, if a monody is being encoded, names such as "e1," "e2," ... are appropriate for music events; if many single-voice parts are present, names could be chosen such as "p1_1," "p1_2," ... and "p2_1," "p2_2," ... ; finally, when the complete hierarchy parts/voices is used, suitable names for music events could be "p1_v1_1," "p1_v2_1," etc.
- `timing` represents the temporal coordinate of the event: it describes the time offset from the occurrence of the preceding event. Time is expressed in virtual time units (VTU), as described in SMDL specifications. When no value can be attributed to a music event, string "null" can be employed. The measurement unit can be fixed according to the user's need, but all the numeric values must be integers. In general, the minimum integer value should be attributed to the smaller note length; this is a recommended choice, however it is not mandatory. For instance, assume a piece in which eights, sixteenths, and thirty-seconds are present; in this case, the unit can be employed to represent thirty-seconds, so that all the other lengths are integer multiples. An exception arises when the minimum length is dotted (in the previous example, dotted thirty-seconds): in this case, the unit should be employed to represent the halved length (sixty-fourths). Finally, when there are irregular groups, all the values for `timing` should be rearranged in order to allow the representation of all lengths through integer values. Since "0" means simultaneity in our relative system, each event is simultaneous to all the following ones having `timing` = "0"; if the event itself presents `timing` = "0", it is also simultaneous to all the preceding events up to the first event with a non-zero value, that event itself included.
- `hpos` refers to a virtual horizontal dimension. The value is expressed in a relative spatial unit called *virtual pixels* (VPX), which provide reference points for the vertical alignment of different symbols in

different voices, parts, or staves. The score is considered in this case from an abstract point of view (real scores will be treated in *Notational* layer). As a consequence, staff systems never generate new lines: there is only one staff system embracing all the symbols on a single line, so that only the horizontal axis is used in establishing vertical alignment. The criteria to define the measurement unit are similar to those employed for *timing*. If music is equally spaced in regard to rhythmic values (meaning that e.g., a double value requires a double space on the staff), *timing* and *hpos* can take the same values. Once again, “null” is supported for events that have no spatial coordinates, and “0” means vertical alignment.

6.2.2 Logically Organized Symbols (LOS)

```
<!ELEMENT los (agogics*, text_field*, metronomic_indication*,
               staff_list, part+, horizontal_symbols?, ornaments?, lyrics*)>
```

Element `los` is the container of several symbolic descriptions of the events occurring in the score. When this element is present, one staff list (element `staff_list`) and at least one music part (element `part`) must be described. The definition of its sub-elements explains further details. Other sub-elements are not mandatory.

6.2.2.1 Agogics

```
<!ELEMENT agogics (#PCDATA)>
<!ATTLIST agogics
  bracketed (no | yes) #IMPLIED
  %spine_ref;>
```

Element `agogics` is provided to express textual tempo and mood indications such as “Allegro,” “Moderato,” etc. (see Figure 3 and Figure 4). The element always has to be linked to a spine event.⁷

Attributes

- `event_ref`, a mandatory attribute to identify the event that `agogics` should be linked to. Two alternatives are possible: a dedicated event can be inserted in spine, or agogic indication can be linked to a simultaneous event of another type (for instance, usually such text indications are aligned to time signatures, so they can share the same spine event).
- `bracketed`, an optional attribute that requires the use of brackets when its value is “yes.”

Examples



Figure 3—Agogics

```
<agogics event_ref="event_01">
  Presto
</agogics>
```

⁷ Note that the DTD contains the string “%spine_ref;”. This is an entity defined in the DTD and used here and in other places to represent the attribute “event_ref IDREF #REQUIRED”.

(Allegro)

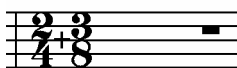


Figure 4—Bracketed agogics

```
<agogics event_ref="event_01" bracketed="yes">
  Allegro
</agogics>
```

6.2.2.2 Text_field

```
<!ELEMENT text_field (#PCDATA)>
<!ATTLIST text_field
  extension_line_to IDREF #IMPLIED
  extension_line_shape (normal | dotted | dashed) #IMPLIED
  %spine_ref; >
```

Element `text_field` contains generic alphanumeric strings, such as “Dolcemente,” “Ritenuato,” or “Staccatissimo” (see Figure 5 and Figure 6). For agogics, the proper element should be used instead (see 6.2.2.1), as well as for dynamics (see 6.2.2.6.5).

Attributes

- `event_ref`, a mandatory attribute to identify the event to which `text_field` should be linked. Two alternatives are possible: a dedicated event can be inserted in spine, or text indication can be linked to a simultaneous event of another type.
- `extension_line_to`, an optional attribute that identifies the last event involved by this text indication. For instance, if “Dolcemente” covers many events, the first one is referred in `event_ref` and the last one in `extension_line_to`. From a graphical perspective, often this indication is realized through an extension line.
- `extension_line_shape`, an optional attribute that sets the type of extension line, if `extension_line_to` is specified.

Examples



Figure 5—Text field

```
<text_field event_ref="event_01">
  pizz.
</text_field>
```



Figure 6—Text field with extension line

```
<text_field event_ref="event_01" extension_line_to="event_07"
           extension_line_shape="normal">
    dolcemente
</text_field>
```

6.2.2.3 Metronomic_indication

```
<!ELEMENT metronomic_indication EMPTY>
<!ATTLIST metronomic_indication
    num CDATA #REQUIRED
    den CDATA #REQUIRED
    dots CDATA #REQUIRED
    value CDATA #REQUIRED
    %spine_ref; >
```

Element `metronomic_indication` allows encoding of metronomic indications, such as $\theta = 60$ (see Figure 7).

Attributes

- `event_ref`, a mandatory attribute to identify the event that `metronomic_indication` should be linked to.
- `num`, a mandatory integer attribute to fix the numerator of the rhythmical value before “=”.
- `den`, a mandatory integer attribute to fix the denominator of the rhythmical value before “=”.
- `dots`, an optional attribute that sets the number of dots for the rhythmical value before “=”.
- `value`, a mandatory integer attribute to fix the number after “=”.

Examples

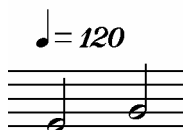


Figure 7—Metronomic indication

```
<metronomic_indication num="1" den="4" value="120" event_ref="ev1" />
```

6.2.2.4 Staff_list

```
<!ELEMENT staff_list (brackets | staff)+>
```

Element `staff_list` provides the list of staves that compose the staff system, i.e., the set of staves to be read and performed together. Staff hierarchy is independent from parts/voices hierarchy; as a consequence, a part can be split on two or more staves, and many parts can share the same staff.

6.2.2.4.1 Brackets

```
<!ELEMENT brackets EMPTY>
<!--ATTLIST brackets
  marker (start_of_staff_group | end_of_staff_group) #REQUIRED
  group_number CDATA #REQUIRED
  shape (square | rounded_square | brace | vertical_bar | none)
-->
```

Element `brackets` describes the type of brackets to create a staff group. Typical examples are the curly bracket used for piano and harp or the symbols used to embrace instrumental groups (e.g., strings) in an orchestral score (see Figure 8). Brackets are expressed through a couple of instances (see the attribute `marker`), in order to embrace all the staves listed inside. In other words, if a number of staves should be grouped by a single bracket, the format requires a start element `brackets` (`marker="start_of_staff_group"`), then a number of staff elements, and finally an end element `brackets` (`marker="end_of_staff_group"`), whose `group_number` is the same of the first element. Obviously, this approach allows inclusion of the same staff in many groups

Attributes

- `marker`, a mandatory attribute to identify if the group is starting or ending here.
- `group_number`, a mandatory attribute to identify univocally the instance of brackets to open/close.
- `shape`, the graphical symbol to employ. Note that also invisible brackets are supported.

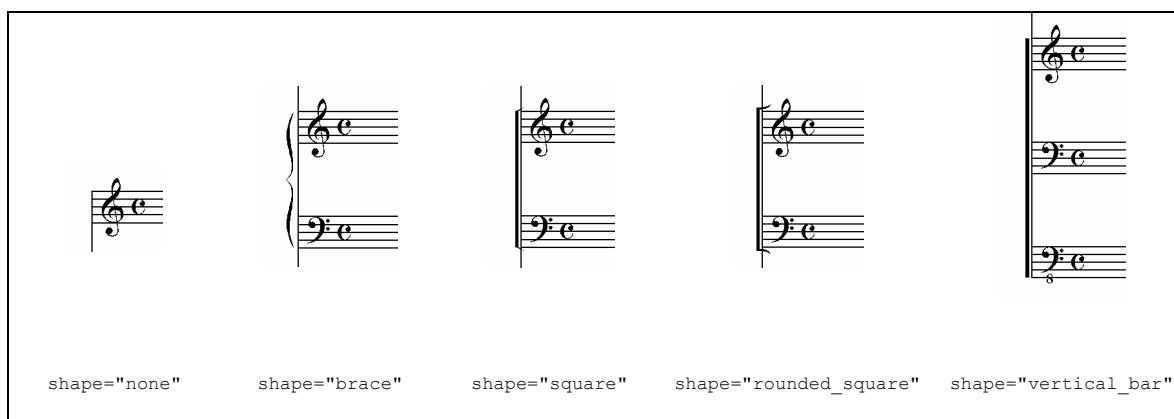


Figure 8—Shapes for brackets

6.2.2.4.2 Staff

```
<!ELEMENT staff (clef | ( key_signature | custom_key_signature) |
  time_signature | barline | tablature_tuning)*>
<!--ATTLIST staff
  id ID #REQUIRED
  line_number CDATA "5"
  ossia (yes | no) "no"
  tablature (none | french | german | italian) #IMPLIED-->
```

Element `staff` describes the characteristics of a single staff. A staff system (see `staff_list`) is made of one staff at least.

Attributes

- `id`, a mandatory attribute that allows referencing to the staff.
- `line_number`, an attribute that contains the number of lines; “5” is the default value.
- `ossia`, an attribute to encode ossia staff. An ossia staff is used as an alternative notation for a part/voice represented in a standard staff. For instance, it is used to realize complex ornaments or to notate standard cadenzas.
- `tablature`, an attribute that states the kind of tablature, if any.

6.2.2.4.2.1 Clef

```
<!ELEMENT clef EMPTY>
<!--ATTLIST clef
  shape (G | F | C | gregorian_F | gregorian_C | percussion |
                                               doubleG | tabguitar) #REQUIRED
  staff_step CDATA #REQUIRED
  octave_num (0 | 8 | -8 | 15 | -15) "0"
  %spine_ref;-->
```

Element `clef` describes the characteristics of the clef. A number of different clefs is supported for the same staff, provided that they are linked to spine events (see Figure 9 and Figure 10). Spine identification allows the identification of the position of clef changes within the score.

Attributes

- `event_ref`, a mandatory attribute to identify the event that element `clef` should be linked to.
- `shape` defines the shape of the clef. Main shapes in use are treble, bass, and C clefs, but percussions, tablature, and double G clefs are also supported by IEEE 1599.
- `staff_step`, an attribute that defines the vertical position of the clef. A NIFF-like syntax has been employed: lines and spaces are numbered from the bottom line, so the value “0” represents the central-E line in a G-clef staff, “1” is the F space, “2” the G line, and so on. This mechanism is an absolute way to identify position with respect to the staff graphical symbol.
- `octave_num`, an attribute to encode numbers over or below the clef symbol. These numbers are used when the musical line is occasionally still too high or too low to fit neatly onto the five-line staff. A useful device that overcomes this problem is a commonly accepted syntax that moves the musical line up or down an octave. In music theory, to make a clef one (two) octave(s) lower, a small 8va (15ma) is added beneath it; likewise, to make a clef one (two) octave(s) higher, a small 8vb (15mb) is added above it. The music is read as though at a certain octave but sounds either one (two) octave(s) higher or one (two) octave(s) lower than it is written.

	<code><clef type="G" staff_step="2" event_ref="c1"/></code>
Treble	
	<code><clef type="C" staff_step="0" event_ref="c1"/></code>
Soprano	
	<code><clef type="C" staff_step="2" event_ref="c1"/></code>
Mezzo-soprano	
	<code><clef type="C" staff_step="4" event_ref="c1"/></code>
Alto	
	<code><clef type="C" staff_step="6" event_ref="c1"/></code>
Tenor	
	<code><clef type="F" staff_step="4" event_ref="c1"/></code>
Baritone	
	<code><clef type="F" staff_step="6" event_ref="c1"/></code>
Bass	
	<code><clef type="tabguitar" staff_step="4" event_ref="c1"/></code>
Tablature	
	<code><clef type="percussion" staff_step="0" event_ref="c1"/></code>
Percussion	

Figure 9—Standard clefs

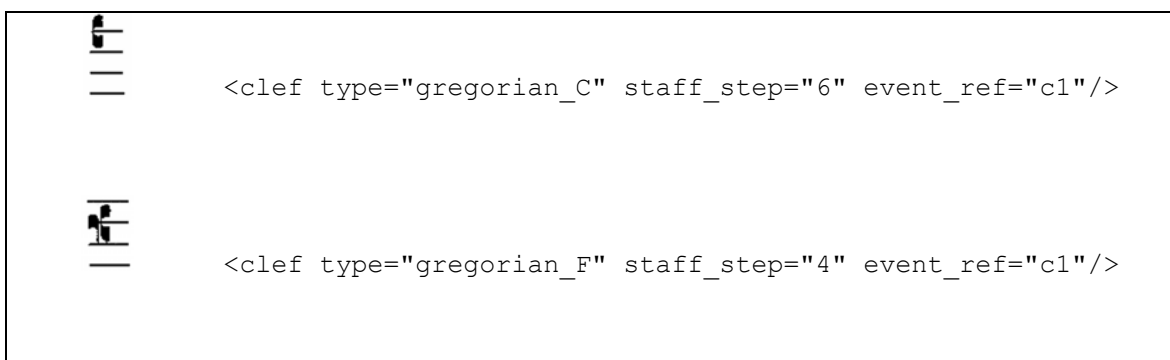


Figure 10—Gregorian clefs

6.2.2.4.2.2 Key_signature and sub-elements

```
<!ELEMENT key_signature (sharp_num | flat_num) >
<!--ATTLIST key_signature
    %spine_ref;-->

<!ELEMENT sharp_num EMPTY>
<!--ATTLIST sharp_num
    number (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7) #REQUIRED>

<!ELEMENT flat_num EMPTY>
<!--ATTLIST flat_num
    number (0 | 1 | 2 | 3 | 4 | 5 | 6 | 7) #REQUIRED>

<!ELEMENT custom_key_signature (key_accidental)+>
<!--ATTLIST custom_key_signature
    %spine_ref;-->

<!ELEMENT key_accidental EMPTY>
<!--ATTLIST key_accidental
    step (A | B | C | D | E | F | G) #REQUIRED
    accidental %accidental; "none">
```

Element `key_signature` and element `custom_key_signature` represent two different ways to encode key signatures (see Figure 11).

The former is used for standard cases: it is sufficient to specify if the key signature uses sharp or flat symbols (sub-elements `sharp_num` and `flat_num` respectively) and their number (mandatory attribute `number`). When a key-signature change is specified, only the new accidentals are encoded and not the number of natural symbols to neutralize the previous accidentals. An exception arises when the new key signature is C major: in this case, it is possible to employ both `number="0"` for `sharp_num` sub-element and `number="0"` for `flat_num`.

The latter method allows the encoding of non-traditional key signatures through a number of `key_accidental` sub-elements (see Figure 12). The attributes of `key_accidental` encode the step to be inflected (attribute `step`) and the kind of accidental to be used for that step (attribute `accidental`).

Examples

The following example illustrates the use of key-signature changes referred to a single staff, namely “staff_01.” Key signatures occur in correspondence with events “ts1,” “ts2,” and “ts3.”



Figure 11—Key signatures

```
<staff_list>
  <staff id="staff_01" line_number="5">
    <key_signature event_ref="ts1">
      <flat_num number="3"/>
    </key_signature>
    <key_signature event_ref="ts2">
      <sharp_num number="2"/>
    </key_signature>
    <key_signature event_ref="ts3">
      <sharp_num number="0"/>
    </key_signature>
  </staff>
  ...
</staff_list>
```

The following example illustrates the use of both a standard key signature and a non-traditional one. Key signatures occur in correspondence with events “ts1” and “ts2.”



Figure 12—Non-standard key signatures

```
<staff_list>
  <staff id="staff_01" line_number="5">
    <key_signature event_ref="ts1">
      <flat_num number="1"/>
    </key_signature>
    <custom_key_signature event_ref="ts2">
      <key_accidental step="F" accidental="doublesharp"/>
      <key_accidental step="C" accidental="sharp"/>
      <key_accidental step="D" accidental="flat"/>
      <key_accidental step="E" accidental="doubleflat"/>
    </custom_key_signature>
  </staff>
  ...
</staff_list>
```

6.2.2.4.2.3 Time_signature

```
<!ELEMENT time_signature (time_indication)+>
<!ATTLIST time_signature
  visible (yes | no) "yes"
  %spine_ref;>
```

Element `time_signature` is a container for a number of elements `time_indication`. In a standard case, only one `time_indication` is required; if more than one are needed, it is possible to encode complex tempo signatures, such as $\frac{1}{2} + \frac{3}{4}$. Mixed meters (e.g., a measure in $\frac{1}{2}$ followed by a measure in $\frac{3}{4}$, alternatively) can be represented by notating all the time signature changes.

Attributes

- `event_ref`, a mandatory attribute to identify the event that element `time_signature` should be linked to.
- `visible`, an attribute that allows hiding the time signature even when music theory would require its visualization.

6.2.2.4.2.3.1 Time_indication

```
<!ELEMENT time_indication EMPTY>
<!ATTLIST time_indication
  num CDATA #REQUIRED
  den CDATA #IMPLIED
  abbreviation (yes | no) "no"
  vtu_amount CDATA #IMPLIED>
```

Element `time_indication` is the most elementary entity to notate simple tempo signatures (see Figure 13, Figure 14, Figure 15, and Figure 16).

Attributes

- `num`, a mandatory attribute to fix the numerator of the fractional representation of tempo. Non-integer values are also allowed.
- `den`, an optional integer attribute to fix the denominator of the fractional representation of tempo. When its value is not specified, the `num` attribute is the only indication to consider and to show.
- `abbreviation`, an attribute that specifies if tempo should be annotated with the alternative character-based notation (when possible). For instance, $\frac{4}{4}$ can be represented as “common” time signature C.
- `vtu_amount` contains the length of the measure in terms of VTUs, according to the conventional values established in spine.

Examples



Figure 13—Standard time signature

```
<time_signature event_ref="event_01">
  <time_indication num="5" den="4" />
</time_signature>
```



Figure 14—Time signature with abbreviation

```
<time_signature event_ref="event_01">
  <time_indication num="4" den="4" abbreviation="yes" />
</time_signature>
```



Figure 15—Single-number time signature

```
<time_signature event_ref="event_01">
  <time_indication num="4" />
</time_signature>
```



Figure 16—Additive meters

```
<time_signature event_ref="event_01">
  <time_indication num="2" den="4" />
  <time_indication num="3" den="8" />
</time_signature>
```

6.2.2.4.2.4 Barline

```
<!--ELEMENT barline EMPTY-->
<!--ATTLIST barline
  style (dashed | double | final | invisible | standard | smaller |
                                                minimum) #REQUIRED
  extension (single_staff | staff_group | all_staves | mensurstrich)
                                                #REQUIRED
  %spine_ref;>
```

Element `barline` is used for the description of different kinds of barline (see Figure 17). Repeated barlines should not be encoded using this element.

Attributes

- `style`, an attribute to specify the graphical aspect of the barline.
- `extension`, an attribute that encodes the number of staves the barline is applied to: `single_staff` for the staff of the current event only, `staff_group` for the group the current event belongs to, `all_staves` for the whole staff system, and `mensurstrich` for lines only between staves.
- `event_ref`, a mandatory attribute to identify the spine event that precedes the barline to be specified. If a single staff is involved (see attribute `extension`), the event identifies the staff to modify. Note that events are associated to parts that are associated to staves. If a group is involved, the group is the one containing the staff of this spine event.

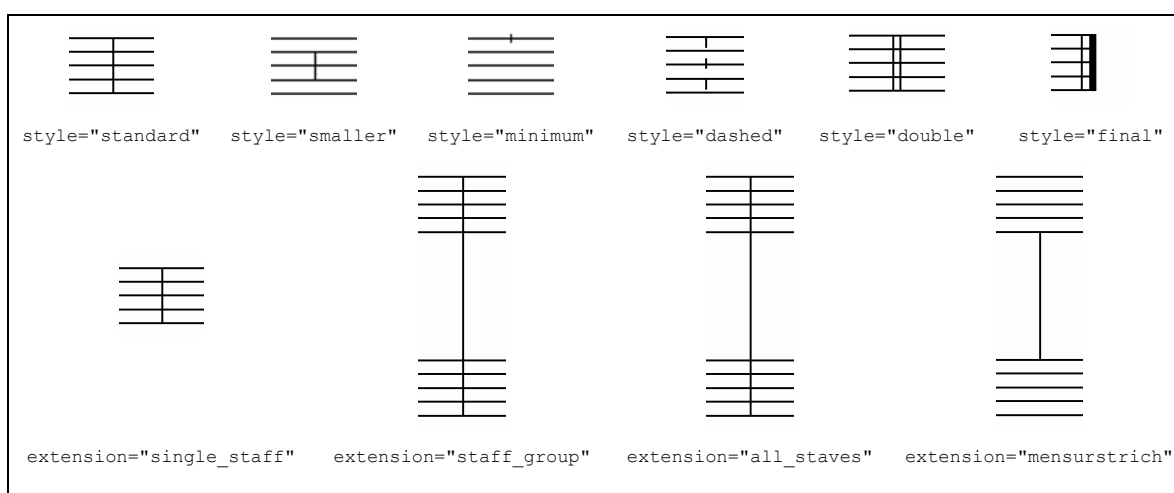


Figure 17—Attributes of barlines

6.2.2.4.2.5 Tablature_tuning

```
<!ELEMENT tablature_tuning (string*)>
<!ATTLIST tablature_tuning
  type (D | E | G | A | baroque | flat_french | other) #IMPLIED>
```

Element `tablature_tuning` describes the tuning for a tablature. Presets can be specified through `type` attribute. When `string` sub-elements conflict with presets, the formers overwrite the latters.

Attributes

- `type`, an optional attribute that specifies a standard tuning.

6.2.2.4.2.5.1 String

```
<!ELEMENT string EMPTY>
<!ATTLIST string
  string_number CDATA #REQUIRED
  string_pitch (A | B | C | D | E | F | G) #REQUIRED
  string_accidental %accidental; #IMPLIED
  string_octave CDATA #REQUIRED>
```

Element string describes the tuning of a string for a tablature.

Attributes

- string_number, the number of the string to be tuned.
- string_pitch, the pitch for the current string.
- string_accidental, the inflecting for the current string.
- string_octave, the octave for the current string.

6.2.2.5 Part

```
<!ELEMENT part (voice_list, measure+)>
<!ATTLIST part
  id ID #REQUIRED
  performers_number CDATA "unknown"
  transposition_pitch (A | B | C | D | E | F | G) #IMPLIED
  transposition_accidental %accidental; #IMPLIED
  octave_offset CDATA #IMPLIED>
```

Element `part` corresponds to the concept of a *part* in a score. For instance, “Violin I,” “Piano,” and “3 flutes” are parts. Internal subdivisions of parts are called “voices” and listed inside `voice_list` sub-element. Music contents are subdivided into a number of parts, then they are described measure by measure.

Attributes

- id, a mandatory attribute to identify univocally the current part.
- performers_number, an attribute to define the number of performers who execute the part.
- transposition_pitch, an optional attribute that specifies the pitch for transposing instruments (e.g., “C” for a trumpet in C, “A” for a clarinet in A, etc.).
- transposition_accidental, an optional attribute that specifies the accidental for transposing instruments.
- octave_offset, an optional attribute that specifies, by a signed integer value, the number of octaves to add (positive values) or to subtract (negative values) in order to identify the correspondence between notated and performed sounds. For instance, even if they cannot be considered transposing instruments, double basses usually perform notes written one octave up: in this case, octave offset is “-1”.

6.2.2.5.1 Voice_list and voice_item

```
<!ELEMENT voice_list (voice_item+)>
<!ELEMENT voice_item EMPTY>
<!ATTLIST voice_item
  id ID #REQUIRED
  staff_ref IDREF #REQUIRED
  notation_style (normal | rhythmic | slash | blank) #IMPLIED>
```

Element `voice_list` is a container for a number of `voice_item` elements. Each part contains at least one voice, but many voices are supported. For instance, in a four voices fugue for organ, the one and only part is organ, which includes four voices. Each voice is identified and described by a `voice_item` sub-element.

Attributes

- *id*, a mandatory attribute to identify univocally the current voice. Note that *id* attribute is referred to in the following by *voice* element, a sub-element of *measure*.
- *staff_ref*, a mandatory attribute to provide a default mapping of a voice on a staff (attribute *id* of element *staff*). This setting can be overridden at notehead level of granularity.
- *notation_style*, an optional attribute that allows definition of alternative notations for a voice.

6.2.2.5.2 Measure

```
<!ELEMENT measure (voice+ | multiple_rest | measure_repeat?)>
<!ATTLIST measure
  number CDATA #REQUIRED
  id ID #IMPLIED
  show_number (yes | no) #IMPLIED
  numbering_style (arabic_numbers | roman_numbers | small_letters |
                  capital_letters) #IMPLIED>
```

Element *measure* represents the concept of bar. Note that musical symbolic contents are listed part by part, measure by measure, and finally voice by voice.

Attributes

- *number*, a mandatory attribute that allows bar numbering.
- *id*, an optional attribute having an alternative identifier for measures.
- *show_number*, an attribute that specifies if the measure number should be hidden or shown.
- *numbering_style*, an optional attribute that defines the style to be used for measure numbering.

6.2.2.5.2.1 Voice

```
<!ELEMENT voice (chord | rest | tablature_symbol | gregorian_symbol)+>
<!ATTLIST voice
  voice_item_ref IDREF #REQUIRED
  ossia (yes | no) "no">
```

Element *voice* contains music symbols, organized voice by voice. Sub-elements encode chords, rests, tablature symbols, or Gregorian symbols.

Attributes

- *voice_item_ref*, a mandatory attribute that refers to the *id* defined in a *voice_item* element.
- *ossia*, an attribute that defines if the voice should be treated as an *ossia*.

6.2.2.5.2.1.1 Chord

```
<!ELEMENT chord (duration, augmentation_dots?, (notehead+ | repetition)
               articulation?)>
<!ATTLIST chord
  id ID #IMPLIED
  %spine_ref;
  stem_direction (up | down | none) #IMPLIED
  beam_before (yes | no) "no"
```