ISO/IEC JTC 1/SC 29/WG 03 N1213

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Technology under Consideration on ISO/IEC 14496-34 Syntactic Description Language

**Status:** Approved

**Date of document:** 2024-05-16

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**No. of pages:** 5 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N1213**

**May 2024, Rennes, FR**

|  |  |
| --- | --- |
| **Title** | **Technology under Consideration on ISO/IEC 14496-34 Syntactic Description Language** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | **23823** |

# Introduction

This document collects technologies being under study for consideration in the development of the standard ISO/IEC 14496-34 Syntactic Description Language

# Table of content

[1. Introduction 1](#_Toc166749844)

[2. Table of content 1](#_Toc166749845)

[3. Grammar file for SDL syntax 1](#_Toc166749846)

[4. On the keyword template 4](#_Toc166749847)

# Grammar file for SDL syntax

## General

This clause provides the SDL grammar implementing the rules of the present specification. The grammar is based on the parsing expression grammar (PEG) concept and following the syntax and set of rules defined by the Pegen software project [1].

Currently, the grammar file is hosted an developed at <http://mpegx.int-evry.fr/software/MPEG/Systems/sdl/sdl-grammar>.

The usage of PEG seems adequate for the validation of the SDL syntax, however there are still some questions whether the PEG grammar defined by the Pegen software project is appropriate for the developing the conformance of the SDL specification. Further study is encourage on other possible alternative for such grammar file.

## Grammar file

[Editor’s note] The following is in work-in-progress and may not reflect all the rules described in the latest SDL specification text.

1. SDL grammar file

|  |
| --- |
| start: file\_input  file\_input: (NEWLINE+ | line)\* ENDMARKER  line: class\_def | comment\_cpp  #NOTE: 14496-1 forbids going back to a new line berfore {  # Rule C.1 and C.2  class\_def: aligned? abstract? 'class' NAME+ parameter\_list? parent\_class? NEWLINE? '{' body? '}'  aligned: 'aligned' '(' NUMBER+ ')'  abstract: 'abstract'  parameter\_list: '(' ','.parameter+ ')'  # NOTE: we allow arrays to be passed as parameter class, however 14496-1 is not clear on this  parameter: optional? type NAME array\_length?  # NOTE: Not in 14496-1, not sure where this come from  optional: 'optional'  # NOTE: would allow "unsigned bit", to be improved  type: signed? data\_type  signed: 'unsigned'  data\_type:  | 'bit'  | 'int'  | 'double'  array\_length: '[' NUMBER\* ']'  #NOTE: 14496-1 does not allow paramters after parent class name  parent\_class: 'extends' NAME '(' ','.value+ ')'  body: stmt\*  stmt:  | elementary\_data\_type  | non\_parsable\_variable  | assignment\_stmt  | object\_instantiation  | increment\_stmt  | if\_stmt  | switch\_stmt  | for\_stmt  | do\_stmt  | while\_stmt  | comment\_cpp  #TODO: See how to do any character up to newline  comment\_cpp: '//' (NAME | 'floor' | 'class' | 'if' | 'else' | 'for' | 'extends' | NUMBER | '==' | '=' | '{' | ';' | ',' | '-' | '/' | ':' | '?' )\*  # Rule E.1 and A.1  elementary\_data\_type: template? aligned? const? type length NAME array\_length? assigned\_value? ';'  non\_parsable\_variable: template? const? type NAME array\_length? assigned\_value? ';'  #NOTE: Not in 14496-1 but used in 14496-12  template: 'template'  const: 'const'  length: '(' (NUMBER | NAME) ')'  #NOTE: array initialisation with {val1, val2, ...} not in 14496-1  assigned\_value: '=' (value | array\_initialisation)  object\_instantiation: NAME NAME ( '(' ','.value+ ')' )\* array\_length? ';'  variable\_assignment: NAME assigned\_value  assignment\_stmt: variable\_assignment ';'  #TODO: This rule should not allow whitepaces between name and '+'s  variable\_incr: NAME '+' '+'  increment\_stmt: variable\_incr ';'  #NOTE: STRING literal e.g. 'uuid' is not allowed in 14496-1  #NOTE: NUMBER catches decimal, octal, hexadecimal, binary, foating point (scientific noation) and even imaginary number. Too broad for SDL.  value: function | expr | '-'? NUMBER | NAME | STRING  expr: (value operator value) | ( '(' value operator value ')' )  #NOTE: & and && not in 14496-1 but used in 14496-12  operator: operator\_test | operator\_logical | operator\_bin | operator\_math  operator\_math: '+' | '-' | '/' | '\*'  operator\_test: '==' | '<=' | '<' | '>=' | '>' | '!='  operator\_bin: '&' | '|'  operator\_logical: '&' '&' | '|' '|'  function: function\_name '(' value ')'  #NOTE: Only lengthof in 14496-1, floor is used in 14496-12 without definition  function\_name: 'floor' | 'lengthof'  array\_initialisation: '{' ','.value+ '}'  # Rule FC.1  if\_stmt: 'if' '(' condition ')' '{' body '}' else\_if\_stmt? else\_stmt?  else\_if\_stmt: 'else' 'if' '(' condition ')' '{' body '}'  else\_stmt: 'else' '{' body '}'  condition: value  # Rule FC.2  switch\_stmt: 'switch' '(' condition ')' '{' (switch\_case switch\_break?)\* switch\_default? switch\_break?'}'  switch\_break: 'break' ';'  switch\_case: 'case' (NUMBER | NAME | STRING) ':' body?  switch\_default: 'default' ':' body?  # Rule FC.3  for\_stmt: 'for' '(' expression1 ';' expression2 ';' expression3 ')' '{' body '}'  for\_variable\_declaration\_assignment: type NAME array\_length? assigned\_value  expression1:  | variable\_assignment  | for\_variable\_declaration\_assignment  expression2: value  expression3: variable\_incr  # Rule FC.4  do\_stmt: 'do' '{' body '}' 'while' '(' condition ')' ';'  # Rule FC.5  while\_stmt: 'while' '(' condition ')' '{' body '}' |

## References

1. Pegen documentation, <https://we-like-parsers.github.io/pegen/>

# On the keyword template

The template keywork in defined in ISO/IEC 14496-12 and does not belong to the original nor currently developped SDL.

The goal of the keyword is to allow other possible values than the one defined for the field by the assignment operator.

EXAMPLE ⎯

template int(32) rate = 0x00010000; // typically 1.0

In this example, the field rate shall be “0x00010000” for a file complying to this specification. But the keyword template allows a derivate spec to define another value.

From the point of a view of a file parser that only knows about the current specification, it shall throw an error is rate as a different value than “0x00010000”.

The ongoing 8ed of ISOBMFF is attempting to clarify the definition and the reader/writer behaviour.

It should be studied whether template should remain an ISOBMFF extension of the SDL or become a feature of the SDL.