Machine-readable and interoperable
age classification labels in Europe

Grant agreement no: 621059

MIRACLE's updated data model for age classification information and electronic labels

MIRACLE specification v2.0

September 24th 2015

Deliverable D4.1

Deliverable Type: Other (data vocabulary & documentation)

Nature of the Deliverable: PU

Editors: HBI, JUSPROG, FSM

Contributors: BBFC, NICAM, PEGI

Contents

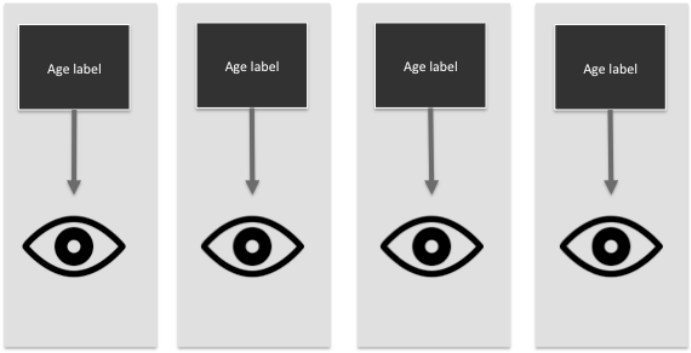
A. Changelog

V2.0	**V1.0 changes**	**Remarks**
xs:schema	*Version changes*	*V1.0 → V2.0*
<label>		**New root element** to enable more than one dataset within one XML file („feeds“)
<scope-title>	New attribute „language“, XSD references external XSD to isolate the volatile language code vocabulary (isolangcodes.xsd)	Introduction of new non-mandatory attribute for <scope-title> that uses two-letter language code to state what language the title is in; ISO 639-1 language codes
<scope-url>	No changes in schema	Reference to RFC4592, extended best practice advices
<scope-api>		Additional scope element referencing an API endpoint as URI; for huge websites with a lot of different classifications, e.g. user generated content platforms
<scope-metadata>; children: <scope-meta class=””>		New element for metadata related to content comprised by the label scope, e.g. duration, release year, producer etc.
<country-code>	No changes in schema, but XSD now references external XSD to isolate the volatile country code vocabulary (isocountrycodes.xml)	Additional regional codes added
<Signature>	Additional element for enveloped XML signatures	Inclusion of W3C xmldsig-core

B. Background and objective

Information about media content is deemed an important aspect for ensuring rational consumer decisions in the digital market. In the field of suitable and unsuitable content for children, many EU Member States rely on age classification procedures and visual age labels. By informing children and parents about relevant media content, such age labels empower them to exercise informed choices regarding media usage, literacy and education.

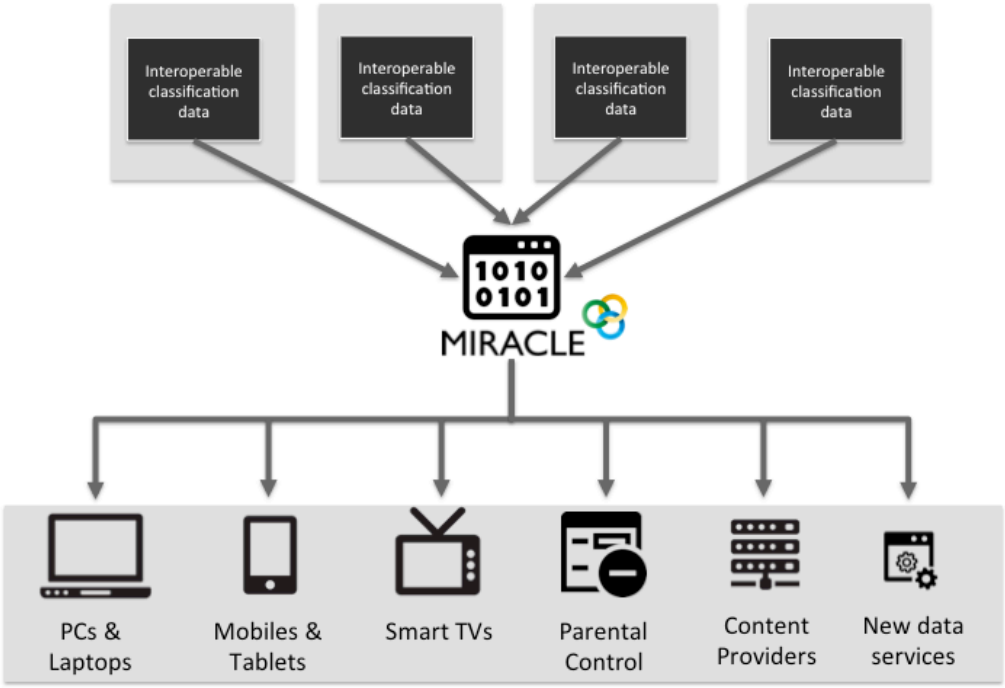
Status quo: National, media- and scheme-related age classification silos



However, age ratings and classification schemes are currently highly fragmented among the Member States and – in most cases – rely on visual labels only. In digital contexts, though, *electronic* age labels open the chance for exchanging and processing classification information in digital realms. Given that one common data model is being used across borders and platforms, such an approach could fully utilise the knowledge comprised in these labels – for the benefit of both end users and businesses along the supply chain and across borders, regions and devices.

Before this background the MIRACLE project ("Machine-readable and interoperable age classification labels in Europe") aims at providing one data model for electronic age classification information, implementing it in different classification schemes and showing the added value of interoperable classification data for businesses, educational institutions and end users. The eight project partners spread across five different member states and classification schemes, consisting of classification bodies, Safer Internet nodes, self-regulatory bodies and filter software providers. The 30-month technical pilot started in Spring 2014 and is co-funded by the "ICT Policy Support Programme" within the CIP ("Competitiveness and Innovation Framework Programme") of the European Union.

Advantages of interoperable age classification data



After an initial draft and a public consultation phase, the project members published Version 1.0 of the MIRACLE specification in autumn 2014. With Version 1.0 the data model took into account current classification schemes and practices while considering existing electronic labelling schemes in order to be easily adapted. Its unrestrictive and open approach leaves enough flexibility for existing and future schemes to map their classification data to MIRACLE. More so, it even enables them to extend the specification to deliver more metadata, if desired. With the actual age label still at its core, the data model also provide fields for content descriptors and – in the field of interactive media content – feature descriptors, e.g. location-based services.

By February 2015, four consortium partners implemented the specification in their very own classification scheme context, offering API endpoints that provide MIRACLE datasets. For the first time, interoperable age labels are available across borders: UK-based BBFC, Netherlands-based NICAM as well as the Pan-European Game Information System PEGI have mapped their existing classification data on the MIRACLE data model and offer MIRACLE-compatible access to (parts of) their classification databases. The German FSM provides an online mapping service that "translates" existing age-de.xml labels into MIRACLE data sets on the fly. Moreover, NCBI as the safer Internet node in the Czech Republic started to pilot a MIRACLE-based database with an API endpoint from summer 2015.

During the implementation phase, valuable experiences and insights have been made by the consortium regarding the overall strategy of implementing an interoperable data model, relevant context factors for opening up classification data to cross-border provision as well as regarding technical aspects of mapping a new specification to existing databases and schemes. Both the decision-making regarding the implementation strategies and the actual implementation steps have been documented by each of the five consortium partners. These reports can be of great value for third

parties when it comes to implementing MIRACLE as well. Based on the experiences made during the implementation as well as on remarks from industry stakeholders regarding additional requirements in electronic age labels the consoritum has agreed on a version 2.0 of the MIRACLE specification in August 2015.

Aiming at becoming a de facto standard for electronic age classification information, MIRACLE put specific efforts in disseminating the pilot project's objectives, the MIRACLE specification as well as general information about the added value of interoperable data in the field of digital child safety. Being a pilot project that is open to and interested in third-party uptake, all relevant stakeholders are invited to provide or use MIRACLE-compatible data. The project members are in constant talks with classification bodies, content producers, distributors, platforms and online services on potential surpluses of using MIRACLE data.

C. MIRACLE's data model for machine-readable age classification data and online labels

Basic principles of the data model

The data model builds on current age labelling practices, as it otherwise would undermine the efforts already taken by companies, rating bodies as well as the classification knowledge that goes with such rating schemes. For companies and classification bodies that already provide classification data or label online content electronically no disadvantage should result from this specification.

This starting point leads to three basic principles of the data model:

(1) The data model will be provided in a **platform-neutral** way to reach maximum openness and compatibility between different systems and languages. Therefore, the specification will propose the data structure, its categories, their fields and the possible attributes of single fields. Examples provided in this document use XML; however, a JSON-focused documentation is being planned.
(2) It **considers existing electronic labelling schemes** to ensure that those approaches are not undermined by the interoperable data model and can easily be mapped onto it.
(3) It **takes into account existing national and supranational classification schemes**. By doing so, existing visual labels can easily be extended by respective electronic labels, ensuring backwards-compatibility with both the data model and the underlying traditional scheme.

Proposing a data model that adheres to these principles aims at fostering a technically interoperable data structure among different rating schemes, technical implementations and distribution contexts of rated content. Information following this specification is machine-readable and can be processed in different software, apps and electronic appliances.

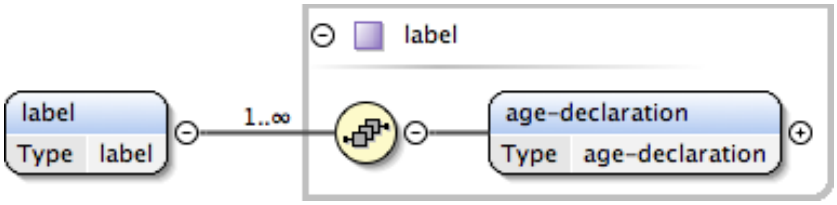
The fundamental principle of the data model is that every scheme only delivers what it can: Neither existing approaches and schemes nor future ones have to provide information in *all* categories provided by the specification. As long as the data bits that are provided by a label can be mapped on any of the proposed categories and fields, the system is technically interoperable. However, the more information a system or label provides the better other IT systems will be able to use and process the data.

During the development of the specification the MIRACLE consortium has decided not to make use of existing metadata schemes, since they either focus on web resource descriptions only or on media metadata in general. While both approaches are fruitful and may be used to extent the MIRACLE specification at a later stage, they do not fit into specific age classification information-related vocabularies of *all* kinds of media, like movies, games, episodes, video clips, apps – and all being *online or offline* resources. Moreover, from the experiences made with former approaches (e.g. ICRA, QUATRO) and current practices (Germany: age-de.xml), the labelling process should be understandable by professional metadata engineers, in-house youth protection or editorial staff as well as by layman content providers. Overly complex specifications for describing web resources, for instance, run into the risk of overstraining implementation efforts in practice.

Additional remark: This scheme is providing primarily a data framework for interoperable age classification data. Hence the scheme might not provide the most efficient way to provide age data for website filtering, for instance. Therefore, there might be different best practices to implement this schema in different use cases. The MIRACLE project consortium is working on best practice examples of implementing the schema as regards different technical contexts (see http://www.miracle-label.eu).

Root element, dataset, main blocks and elements of the MIRACLE specification

MIRACLE data uses the root element <label>. Within this root element one or several (or many) classification datasets can be delivered. Each of those single dataset is enclosed in <age-declaration> brackets.

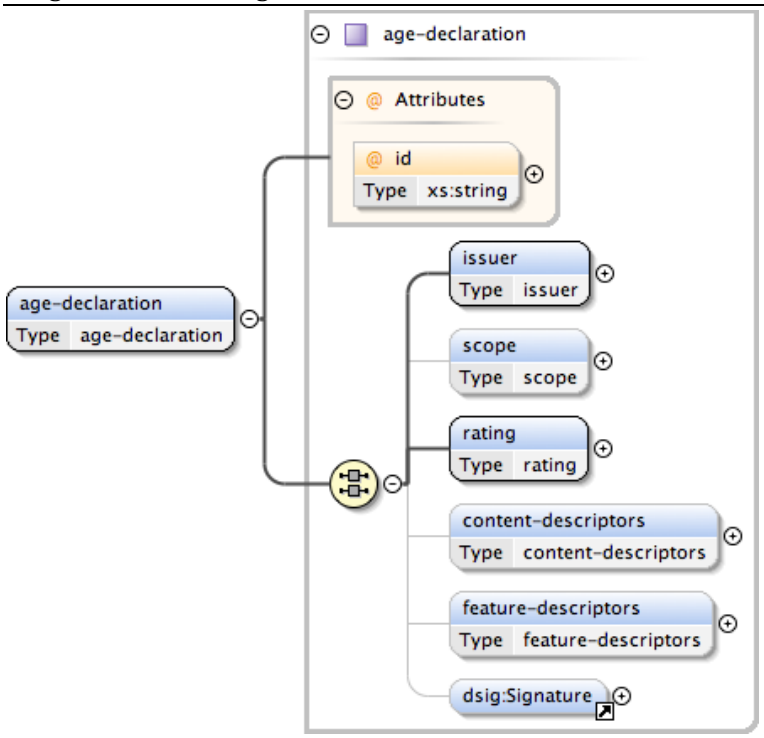



```
<label>
  <age-declaration>
  …
  </age-declaration>
</label>
```

Each of an <age-declaration> consists of several blocks of elements. The main building blocks of <age-declaration> datasets are:

- The body issuing the classification (<issuer>),
- the scope of the dataset (<scope>),
- age labels (<rating>),
- content descriptors (<content-descriptors>),
- feature descriptors (<feature-descriptors>), and
- an enveloped XML signature.

Diagram data set <age-declaration>



General data set structure in XML

```
<age-declaration>
  <issuer>…</issuer>
  <scope>…</scope>
  <rating>…</rating>
  <content-descriptors>…</content-descriptors>
  <feature-descriptors>…</feature-descriptors>
  <Signature>…</Signature>
</age-declaration>
```

Blocks are not in fixed order [xs:all].
Required: <issuer>, <rating>.
Optional: <scope>, <content-descriptors>, <feature-descriptors>, <Signature>.

<age-declaration> may have an attribute "id" in case of multiple datasets regarding the same content, e.g. in case a content comes with more than one data set (e.g. by different issuing bodies or different country scopes).

Block: Issuing body - <issuer>

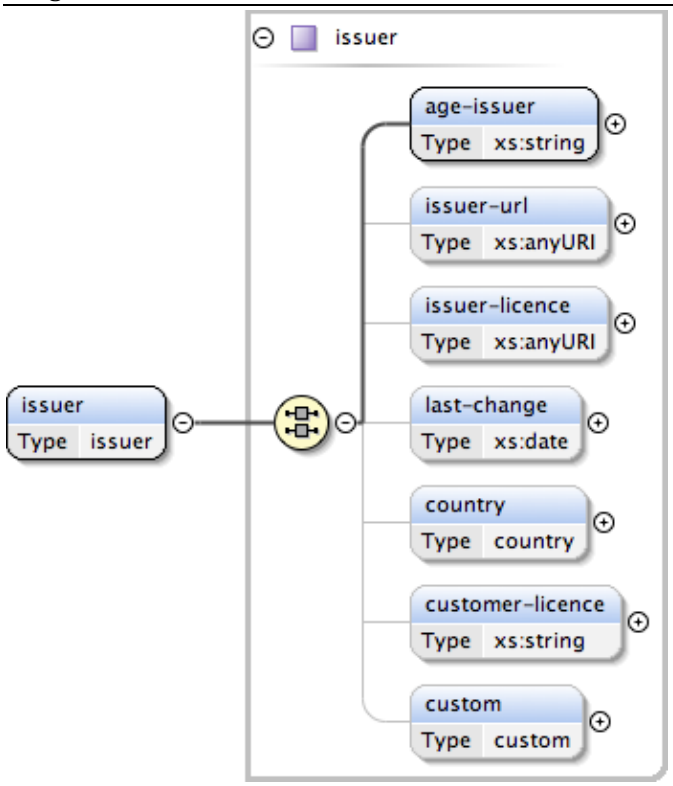
It is important for all content ratings to refer to the body or creator that issued the specific information/label. First, this information links the age label and descriptors to brand/institutional trust. Second, this block allows the provision of the regional origin of the label to assess its individual relevance. By providing the date of the last revision of the classification the label can also show its actuality and help in processing ex post changes.

Elements of <issuer> and children do not have to be provided in a fixed order [xs:all].

Name	**Possible values**	**Comments**	**Mandatory?**
<age-issuer>	shortText [xs:string]	Since age classification can be based on self-classification, this field is open for all kinds of issuing bodies (e.g. PEGI, NICAM, FSM, FSK, „own", „Company Name", "community-based" etc.).	***yes***
<issuer-url>	URL [xs:anyURI]	Reference to web address of issuing body, where additional information on institutional background and rating procedures might be found.	no
<issuer-licence>	URL [xs:anyURI]	URL with link to the issuer's licence regarding the classification information, i.e. the type of licence under which the data is being provided (e.g. copyright,	no

		creative commons, public domain).	
<last-change>	Date (yyyy-mm-dd) [xs:date]	The date of the most recent decision or update regarding the classification information (e.g. 2015-08-28).	no
<country> children: <country-code>	Two-letter ISO country code (upper case); special value for worldwide application (all). [xs:string]	Use country abbreviation according to outsourced isocountrycodes.xsd (e.g. DE, NL, GB, etc.; worldwide: all; multiple countries use child-tag <country-code> for each country. This tag informs about region(s) the label comes from and/or is (not necessarily legally) applied to.	no
<customer-licence>	shortText [xs:string]	Sometimes classifiers' customers provide data under the licence of a classification body; in such cases, this element can provide information regarding the licensee, e.g. a customer ID, a licence number, etc.	No
<custom> children: <custom-field class="">	shortText [xs:string]	Open text field element that the issuing body can freely use for various reasons; the class-attribute is obligatory in case a <custom-field> is used; it can be used both for internal reasons as well as for expressly pointing out underlying schemes or procedures.	no

Diagram – block <issuer>



Block snippet from an exemplary dataset in XML

```
<issuer>
   <age-issuer>BBFC</age-issuer>
   <issuer-url>http://www.bbfc.co.uk</issuer-url>
   <issuer-licence>
      http://www.bbfc.co.uk/sites/default/files/attachments/
      BBFC%20Digital%20Licence_0.pdf
   </issuer-licence>
   <last-change>2013-05-07</last-change>
   <country>
     <country-code>GB</country-code>
   </country>
   <customer-licence>LN-023-2004</customer-licence>
</issuer>
```

Block: Label metadata - <scope>

Depending on the existing form of age classification data (a central classification database that can be queried by third parties or a specific age label "attached" to one specific media content), each piece of interoperable classification information has to offer information about the content the classification information applies to („scope"). This scope might either aim at specific, isolated media content that – sometimes in a nation- or region-specific version – has already been classified or it encompasses a whole website or service, or parts thereof.

Elements of <scope> and their children do not have to be provided in a fixed order [xs:all]. Depending on the context a data set might only contain one of the available scope element types; however, if needed, several types can be used in parallel (e.g. scope-title and scope-id elements). In such cases, each type is defining the scope of the label independently of each other.

Name	**Possible values**	**Comments**	**Mandatory?**
<scope-urls> children: <scope-url class=””>	FQDN, optionally making use of asterisks/“*”, where *. is a wildcard for subdomains (third and higher level domains), covering complete second and higher level domains (*.example.com) “/*“ after the FQDN means label applies to all URLs within that folder hierarchy [xs:string]	FQDN-/URL-based scope of application of a label; the class-attribute can be used both for internal reasons as well as for expressly pointing out the underlying scheme. URL covers all protocols (e.g. http/https). No protocol-information has to be given. More than one scope-url is allowed.	No
<scope-ids> children: <scope-id class=””>	Alphanumeric [xs:string]	UID of classified content; the class-attribute can be used both for internal reasons as well as for expressly pointing out the underlying scheme. More than one scope-id element is allowed.	No
<scope-titles> children: <scope-title class=””	shortText [xs:string]	Title of classified content (system-specific). More than one scope-	No

language=””>		title element is allowed. The class-attribute can be used both for internal reasons or for expressly pointing out the underlying scheme. The language attribute can contain the two-letter language code to state the title's language.	
<scope-hashes> children: <scope-hash class=””>	32-bit hash value	Sometimes an easy way of describing a scope is to refer to an object's hash value. The class-attribute can be used for internal declarations or for expressly pointing out the hash scheme.	No
<scope-api>	URL [xs:anyURI]	In some cases the wildcard-based <scope-url> approach won't suffice to label online content due to highly dynamic and versatile content under one FQDN. Such content providers may offer an API endpoint that delivers full MIRACLE datasets for specific queries.	No
<scope-metadata> children: <scope-meta class=””>	shortText [xs:string]	If a content provider wants to give more information about the content a dataset applies to	No

		he may do so by using one or more <scope-meta> elements, where the class-attribute defines the value's meaning, e.g. <scope-meta class="Runtime">	

The scope-elements provided in this block will usually depend on the context of the label's implementation:

a) Age ratings are traditionally being provided by classification bodies, storing all their rating decisions in *central databases*. Therefore, existing rating bodies will usually opt for a database-driven provision of classification information. The issue here is that for instance an online shop or a VOD service provider needs to query the database for a valid age classification information. As the label is detached from the content, the database query has to aim at getting the correct information back from the database. To receive the correct result both the data provider as well as the demander will either have to use unique identifiers or at least fuzzy approximations by using an object title or more media-related metadata. In practice, such UIDs are not being used coherently throughout all existing rating bodies – and even if a rating body uses UIDs, they aren't the same among different rating bodies. Hence, an alternative can be to base queries on the title of the media content (e.g. movie title, game title). This fuzzy approach might, however, lead to several results, as in many cases different versions of a game or film have been classified by a rating body (cinematic version, DVD version, Director's cut, uncut version, TV version etc.).

To remain flexible, the data model opts for offering several fields for scope-related metadata: a scope URL, an UID field, a title field, or a hash-based value. These fields are not mandatory and can be used either alone or in combination. As each of the field can have one attribute, each issuing body will be able to attach system-specific UIDs to a dataset; e.g. country, rating system, producer, internal ERP or logistics numbers, EAN, GSDN, etc.). Especially in these contexts, providers of classification information will feel the need to provide more metadata regarding the media content the age labels applies to, e.g. specific platforms of distribution, duration of a movie or version of an app or a game. The <scope-metadata> element with its <scope-meta>-fields can therefore contain additional metadata.

b) In contrast to static, retail or offline content, an URI-based scope will always be necessary to clarify the scope of a classification for online content. Moreover, in online contexts the webmaster might be able or willing to give age-related information for his whole website. Here, one age label applies to the whole website or parts thereof and not necessarily one specific URI. Main focus of the <scope-url> element therefore is to offer options to label online content on different levels of abstraction. The basic principle is to take an age label for the whole URI authority, usually the second level domain, and providing possibilities for defining URI-based scopes (specific subdomains, files or folders).

<scope-url> wildcards

For FQDN and URLs, the following wildcards are defined:

(1) Subdomains: *.example.com covers all subdomains including www.example.com, sub1.sub2.example.com, and also example.com without a subdomain. This approach is generally compatible with IETF's RFC4592 document. However, if you want to strictly follow RFC4592 you would have to use two <scope-url> elements, namely *.example.com and example.com.
To label specific subdomains only, you might use *.sub1.example.com with similar rules. Wildcards for whole second level domains are not allowed (e.g. *.eu or *.de are not valid).

(2) Paths, i.e. URI-based scopes behind the first slash:
www.example.com only covers the main page, nothing else;
www.example.com/* includes all URLs of www.example.com, including the main page and e.g. www.example.com/folder1/index.php;
www.example.com/folder1/* covers all URLs inside this folder like www.example.com/folder1/pic1.jpg or www.example.com/folder1/index.php?par=1, including www.example.com/folder1.

Different protocols do not lead to different scopes: https://www.example.com has the same scope as http://www.example.com (both: www.example.com or with wildcard *.example.com).

Temporary exemption for interpreting existing age-de.xml labels

These rules apply to all MIRACLE labels. For existing age-de.xml labels in Germany, please be advised that the interpretation of scopes currently follows these rules:

(1) Paths: age-de labels using www.example.com as their scope *apply to all URLs* of www.example.com, e.g. www.example.com/folder1/index.php. www.example.com/folder1 respectively covers all URLs inside this folder including the URL www.example.com/folder1 itself.

(2) The labelling of the main page with an age rating that differs from the rest of a website/subdomain is currently not possible in age-de.xml. The age-de.xml label specification will be updated in mid-term according to the MIRACLE specification. To provide an age-de.xml label for a whole website that is fully processable by both age-de and MIRACLE based interpreters, please use

<scope>*.example.com/*<scope>
<scope>example.com/*</scope>
<scope>*.example.com</scope>

(Labels regarding specific folders apply respectively.)

Examples for <scope-url>

Whole content of website at http://www.example.com (including all subdomains, example.com and the main page)	`<scope-url>`
	`*.example.com/*`
	`</scope-url>`

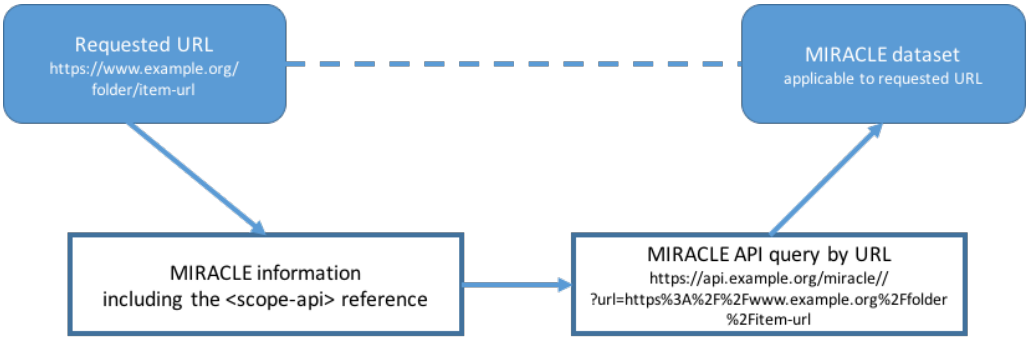
Whole content of website at	`<scope-url>`

http://www.example.com (but not any other subdomains than www.)	www.example.com/* </scope-url>
Whole content of subdomain http://sub1.example.com (including all further level subdomains like http://sub2.sub1.example.com but not www.example.com)	<scope-url> *.sub1.example.com/* </scope-url>
Content of website folder at http://www.example.com/folder1/	<scope-url> www.example.com/folder1/* </scope-url>
Content of website folder at http://subdomain.example.com/folder1/	<scope-url> subdomain.example.com/folder1/* </scope-url>
Content of a page within website folder at http://www.example.com/folder1/example.html	<scope-url> www.example.com/folder1/example.html </scope-url>
Content of the main page as well as one page within website folder at http://www.example.com/folder1/example.html	<scope-url> www.example.com </scope-url> <scope-url> www.example.com/folder1/example.html </scope-url>
Only the content of the main page of the website at http://www.example.com	<scope-url> www.example.com </scope-url>

Referencing a MIRACLE API endpoint <scope-api>

In specific environments, e.g. platforms for user-generated content or large-scale video content providers (e.g. video on demand platforms) the variety of age classifications is too high to provide all labels in one giant MIRACLE dataset or to fall back to the wildcard-based approach. For these occasions the specification provides the opportunity to reference the specific URI of an API endpoint that can be queried and that provides full based MIRACLE datasets based on the specific query. In cases a dataset points to an API endpoint for individual classification information, the mandatory <age> element is considered the fallback age, while the API result is considered the superseding individual dataset.

Referencing a <scope-api>



Providing additional metadata in <scope-metadata>

Sometimes providers of age classification information need to provide additional information regarding the media content a specific label applies to. Such media-related metadata is commonly used in practice, there is, however, no common vocabulary for such metadata that the MIRACLE specification could reference. If a provider of MIRACLE datasets wants to provide additional metadata he may use the <scope-metadata> with ist children <scope-meta>, where the „class"-attribute describes the category of given meatadata. Standards will depend on the type of mediat he MIRACLE dataset applies to.

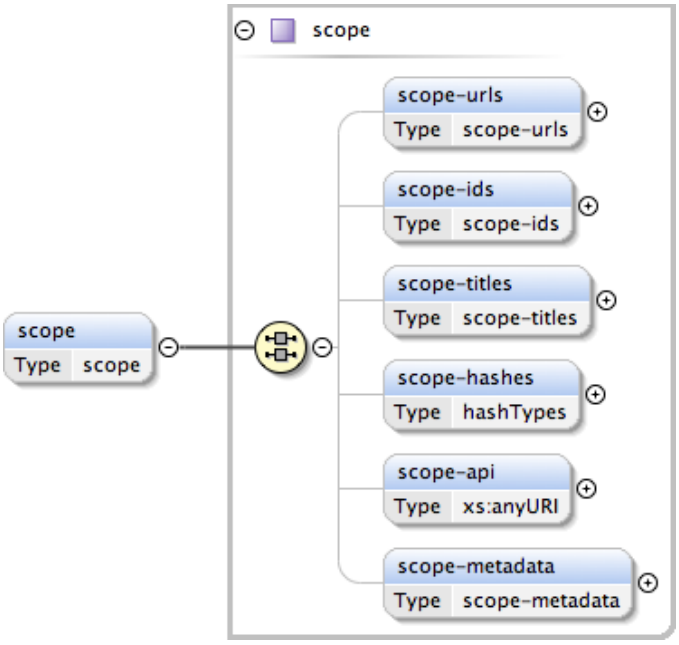
<scope-meta> examples for movies/ TV episodes:

```
Year of release - <scope-meta class="ReleaseYear">
Original title  - <scope-meta class="OriginalTitle">
Runtime         - <scope-meta class="Runtime">
Episode         - <scope-meta class="Episode">
Name of series  - <scope-meta class="SeriesName">
Season          - <scope-meta class="Season">
Episode no      - <scope-meta class="Episode">
Genre           - <scope-meta class="Genre">
Director        - <scope-meta class="Director">
Writer          - <scope-meta class="Writer">
Actors          - <scope-meta class="Actors">
Plot            - <scope-meta class="Plot">
IMDB ID         - <scope-meta class="IMDB-ID">
```

<scope-meta> examples for Games/Apps:

```
Year of Release - <scope-meta class="ReleaseYear">
Platform        - <scope-meta class="Platform">
Publisher       - <scope-meta class="Publisher">
Developer       - <scope-meta class="Developer">
Version         - <scope-meta class="Version">
```

Diagram – block <scope>



Snippet from an exemplary dataset in XML (Note: For documentation purposes only, the example includes different scope types that might not make sense in real life.)

```
  <scope>
    <scope-urls>
      <scope-url class="web-url">*.example.org/halo3/*</scope-url>
    </scope-urls>
    <scope-ids>
      <scope-id class="BBFC-codenumber">AZG238100</scope-id>
      <scope-id class="EAN">9783125171341</scope-id>
    </scope-ids>
    <scope-titles>
      <scope-title class="BBFC-title" language=”en”>HALO 3</scope-
title>
    </scope-titles>
    <scope-metadata>
      <scope-meta class=”Version”>1.1</scope-meta>
      <scope-meta class=”Platform”>Xbox 360</scope-meta>
      <scope-meta class=”Publisher”>Microsoft</scope-meta>
      <scope-meta class=”ReleaseYear”>2007</scope-meta>
    </scope-metadata>
  </scope>
```

Block: Age label - <rating>

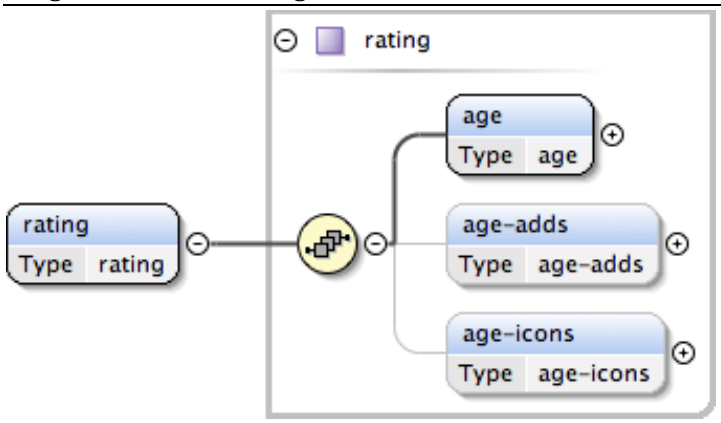
Age labels are a common approach in content rating systems worldwide – age ratings hence are the core aspect of interoperable rating information. However, there are different schemes of how to provide age-specific information (specific age, age group, age group description, or additional age information like parental guidance). As numeric values can be processed by machines in an optimal way compared to open text fields, it is deemed most feasible to translate textual age classification like „teens“ or „mature“ into numbers on side of the classification scheme (and its API) rather than to translate textual values in the data model into number on processor-/client-side. The data model will encompass these differences by making the numeric age field mandatory, but provides options to extend the data structure by additional age-related data fields.

Elements of <scope> and children do not have to be provided in a fixed order [xs:all].

Field	Possible values	Comments	Mandatory?
<age>	Numeric [xs:integer] min=-1 max=99	Minimum age for that the content is deemed suitable. Most age classification systems already use numeric values. However, systems that do not will have to provide translation tables to provide numeric values here (e.g. US ESRB: E→0; E10+→10; T→13; M→17; A→18). Schemes that do not include any age classification (e.g. just content or feature descriptors) should use the attribute “class” with “na” and “-1” as a value. Each dataset must only have one <age> element. If a dataset applies to content (e.g. a website) that consists of categories with different age ratings the label will have to provide two <age-declaration> datasets accordingly.	Yes
<age-adds> children: <age-add class=””>	shortText [xs:string]	If a system uses additional values to specify an age classification, this additional age information can be provided here, e.g. PG, R18, 12A, X etc. Systems that rely on non-numeric classification (e.g. ESRB) can provide their original rating in this field, too. By providing a class attribute the data structure is able to provide a scheme-consistent explanation. More than one <age-add> element is allowed.	No
<age-icons> children:	URL [xs:anyURI]	To provide trustful classifications, this field provides the URL to the original age	No

<age-icon class=””>		rating icon that can be used for displaying an age rating icon if requested. Protocol (e.g. http://) has to be included. More than one <age-icon> element is allowed.	

Diagram – block <rating>



Snippet from an exemplary dataset in XML

```
<rating>
   <age>12</age>
   <age-adds>
    <age-add class="PEGI-age">12+</age-add>
   <age-adds>
   <age-icons>
      <age-icon class="PEGI-icon">http://pegi.eu/label/12.png
      </age-icon>
   <age-icons>
</rating>
```

Block: Content descriptors - <content-descriptors>

Several existing classification schemes use content descriptors to give additional information about the content and the reasons for a specific age rating. The categories of these descriptors are quite comparable up to now. However, there are and always will be peculiarities of single systems or schemes, showing the need for a flexible approach. The data model provides a mixed data field model here, where common and agreed categories are predefined, while the specification permits additional content descriptor elements for the sake of flexibility. In this category, too, icons are commonly used for visualising the respective content descriptors. The data model provides elements for URL references to such icons.

If a pre-defined content descriptor is applicable, the allowed values are true/false only. In this context it doesn't matter if the issuing body considers the respective relevant content as mild or strong, as long as it is relevant for the age classification decision. All content descriptor fields are optional, since some existing schemes do not provide content descriptors at all. Hence if a field is not provided by a data set, *this does not default to "false"*.

Elements of <content-descriptors> do not have to be provided in a fixed order [xs:all]. As being said, the following predetermined categories are not slimming down the options of possible content descriptors; each scheme might introduce its own additional sets of descriptors. However, to achieve a small set of best practice categories, schemes using MIRACLE are strongly encouraged to map their descriptors to the following ones (and add their scheme-specific fields as additional ones where deemed necessary). This way MIRACLE will be able to make different classification schemes technically interoperable.

Pre-defined elements for content descriptors (standardised content descriptors)

Name	**Possible values**	**Comments**
<cd-sexuality>		
<cd-sexuality-exist>	true/false [xs:boolean]	Sex/eroticism/nudeness is a reason for the age classification
<cd-sexuality-desc>	shortText [xs:string]	Description of content
<cd-sexuality-icon>	URL [xs:anyURI]	Address of original sex/erotic/nudeness icon
<cd-violence>		
<cd-violence-exist>	true/false [xs:boolean]	Violence/weapons/ blood is a reason for the age classification
<cd-violence-desc>	shortText [xs:string]	Description of content
<cd-violence-icon>	URL [xs:anyURI]	Address of original violence/weapons/blood icon
<cd-discrimination>		
<cd-discrimination-exist>	true/false [xs:boolean]	Discrimination/ racism/ hate speech is a reason for the age classification
<cd-discrimination-desc>	shortText [xs:string]	Description of content
<cd-discrimination-icon>	URL [xs:anyURI]	Address of original discrimination/ racism/ hate speech icon
<cd-cursing>		
<cd-cursing-exist>	true/false [xs:boolean]	Obscene language/bad language/cursing is a reason for the age

		classification
<cd-cursing-desc>	shortText [xs:string]	Description of content
<cd-cursing-icon>	URL [xs:anyURI]	Address of original obscene/bad language/cursing icon
<cd-drugs>		
<cd-drugs-exist>	true/false [xs:boolean]	Drugs/tobacco/alcohol is a reason for the age classification
<cd-drugs-desc>	shortText [xs:string]	Description of content
<cd-drugs-icon>	URL [xs:anyURI]	Address of original drugs/tobacco/alcohol icon
<cd-fear>		
<cd-fear-exist>	true/false [xs:boolean]	Fear/shock is a reason for the age classification
<cd-fear-desc>	shortText [xs:string]	Description of content
<cd-fear-icon>	URL [xs:anyURI]	Address of original fear/shock icon
<cd-gambling>		
<cd-gambling-exist>	true/false [xs:boolean]	Gambling is a reason for the age classification
<cd-gambling-desc>	shortText [xs:string]	Description of content
<cd-gambling-icon>	URL [xs:anyURI]	Address of original gambling icon

If additional relevant content categories emerge or a scheme would like to provide additional content descriptors than the ones pre-defined, it might do so in the following way within the block <cd-other>.

Example for additional content descriptors

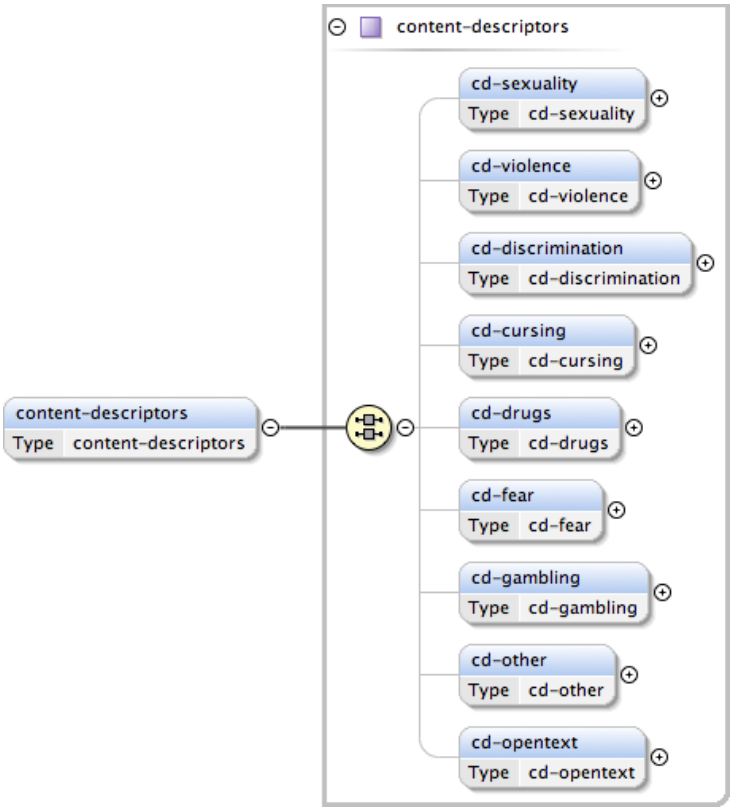
Field name	**Possible values**	**Comments**
<cd-add class="cd-xyz"> (e.g. class="cd-selfharm"; class="cd-antisocial")		The class-attribute is mandatory here, as it contains information about the additional descriptor.

<cd-add-exist>	true/false [xs:boolean]	
<cd-add-desc>	shortText (short description of additional content descriptor) [xs:string]	
<cd-add-icon>	URL; address of original additional icon [xs:anyURI]	

Some systems do not provide systematic content descriptors, but offer additional information regarding the reasoning for a specific age rating in text form. Such information is harder to structure and to process technically but it still provides relevant information. Later on, systems might be able to process this data automatically, too.

Field name	**Possible values**
<cd-opentext>	
<cd-opentext-desc class=””>	shortText [xs:string]

Diagram – block <content-descriptors>



Snippet from an examplary dataset in XML

```
<content-descriptors>
      <cd-violence>
          <cd-violence-exist>true</cd-violence-exist>
          <cd-violence-icon>http://pegi.eu/label/violence.png
          </cd-violence-icon>
      </cd-violence>
      <cd-fear>
          <cd-fear-exist>true</cd-fear-exist>
          <cd-fear-icon>http://pegi.eu/label/fear.png
          </cd-fear-icon>
      </cd-fear>
      <cd-other>
          <cd-add class="self-harm">
              <cd-add-exist>true</cd-add-exist>
              <cd-add-icon>http://pegi.eu/label/self-harm.png
              </cd-add-icon>
          </cd-add>
      </cd-other>
          <cd-opentext class="PEGIonline">
              <cd-opentext-desc>Online game</cd-opentext-desc>
          </cd-opentext>
  </content-descriptors>
```

Block: Feature descriptors - <feature-descriptors>

A different type of descriptors may relate to information about features or functionalities that the content (or the content-related service or platform) provides to

the user. Depending on the scheme, this information sometimes results in a specific age classification result, sometimes it is only regarded as additional information for end users without an effect on the actual age rating decision. Information regarding such features are relevant for minors and other consumers, too; e. g. user generated content might contain relevant depictions that would change existing age classifications, chat functionalities result in unknown people approaching (underage) user in a harmful way or location-based services log and display the movement and/or other person-related information to third parties. For instance, the PEGI scheme already started to implement such descriptors. Hence a first step is to take those as predefined ones, while leaving the feature descriptor field open to new ones, too (see above). All feature descriptor fields are optional, since most existing schemes do not provide content descriptors. If a field is not provided by a data set, *this does not default to "false"*.

Elements of <feature-descriptors> and children do not have to be provided in a fixed order [xs:all]. The following categories are not slimming down the options of possible feature descriptors; each scheme might introduce its own sets of descriptors here. However, to achieve a small set of best practice categories, schemes using MIRACLE are encouraged to map their descriptors to the following ones (and add their scheme-specific fields as additional ones, where deemed necessary; see above).

Pre-defined data fields regarding features (standardised feature descriptors)

Name	**Possible values**	**Comments**
<fd-inapppurchase>		
<fd-inapppurchase-exist>	true/false [xs:boolean]	The service contains elements enabling the consumer to purchase additional content or functionality, regardless of whether the app itself was acquired for free or not.
<fd-inapppurchase-desc>	shortText [xs:string]	Description of feature
<fd-inapppurchase-icon>	URL [xs:anyURI]	Address of original icon for in-app purchase features.
<fd-personaldatasharing>		
<fd-personaldatasharing-exist>	true/false [xs:boolean]	The service gives its provider (or a third party) access to personal data such as home address, contact details or bank account numbers.
<fd-personaldatasharing-desc>	shortText	Description of feature

	[xs:string]	
<fd-personaldatasharing-icon>	URL [xs:anyURI]	Address of original icon for personal data sharing features.
<fd-locationdatasharing>		
<fd-locationdatasharing-exist>	true/false [xs:boolean]	The service contains the option to share exact location on a map when using the service. The location information may be shared publicly or with a specific network inside the service or beyond.
<fd-locationdatasharing-desc>	shortText [xs:string]	Description of feature
<fd-locationdatasharing-icon>	URL [xs:anyURI]	Address of original icon for location data sharing features.
<fd-chat>		
<fd-chat-exist>	true/false [xs:boolean]	The service includes an option for a user to chat with other users via the app. These users may operate under a pseudonym or anonymously.
<fd-chat-desc>	shortText [xs:string]	Description of content
<fd-chat-icon>	URL [xs:anyURI]	Address of original icon for chat features.

Additional feature descriptors will emerge during time. Hence, the specification has to be open to additional or new descriptors, too.

Example for additional feature fields (additional feature descriptors)

Field name	**Possible values**	**Comments**
<fd-add class=”fd-xyz”> (e.g. class=”fd-upload”)		The class-attribute is mandatory here, as it contains information about the additional

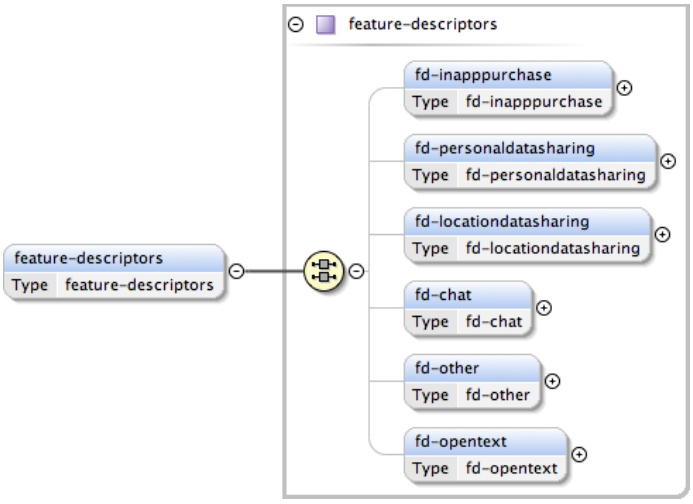
		descriptor
<fd-add-exist>	true/false [xs:boolean]	
<fd-add-desc>	shortText (short description of additional feature descriptor) [xs:string]	
<fd-add-icon>	Address of original additional icon [xs:anyURI]	

Such additional fields might become pre-defined fields in later versions of the data model, when deemed necessary.

Some systems do not provide systematic feature descriptors but offer additional information in text form. Such information is harder to structure and to process technically but it still provides relevant information (see above).

Field name	**Possible values**
<fd-opentext>	
<fd-opentext-desc class=”fd-xyz”>	

Diagram – block <feature-descriptors>



Snippet from an exemplary dataset in XML

```
    <feature-descriptors>
        <fd-inapppurchase>
            <fd-inapppurchase-exist>true
            </fd-inapppurchase-exist>
            <fd-inapppurchase-icon>http://pegi.eu/label/iap.png
            </fd-inapppurchase-icon>
        </fd-inapppurchase>
        <fd-personaldatasharing>
```

```
            <fd-personaldatasharing-exist>true
            </fd-personaldatasharing-exist>
            <fd-personaldatasharing-icon>
            http://pegi.eu/label/pds.png
            </fd-personaldatasharing-icon>
        </fd-personaldatasharing>
        <fd-chat>
            <fd-chat-exist>1</fd-chat-exist>
            <fd-chat-icon>http://pegi.eu/label/chat.png
            </fd-chat-icon>
        </fd-chat>
           <fd-other>
            <fd-add class="self-harm">
                <fd-add-exist>true</fd-add-exist>
                <fd-add-icon>http://pegi.eu/label/self-harm.png
                </fd-add-icon>
            </fd-add>
        </fd-other>
           <fd-opentext>
                <fd-opentext-desc class="PEGI-fd-info">
                Text about additional features
                </fd-opentext-desc>
           </fd-opentext>
    </feature-descriptors>
```

Block: XML Signature - <dsig:Signature>

Using the W3C dsig-core schema the MIRACLE specification is able to provide encrypted or signed XML datasets. Using enveloped signatures, where the <Signature>-block is part of the signed XML file, dsig-core works by computing the digest value over the whole XML document *excluding* the <Signature> element. After that the signature of the SignedInfo element containing this DigestValue can be computed. For further documentation please read http://www.w3.org/TR/xmldsig-core/

Snippet from a signed exemplary dataset in XML

```
<dsig:Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <dsig:SignedInfo>
    <dsig:CanonicalizationMethod
     Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <dsig:SignatureMethod
     Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <dsig:Reference URI="">
      <dsig:Transforms>
        <dsig:Transform
         Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
         signature" />
      </dsig:Transforms>
      <dsig:DigestMethod
       Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
     <dsig:DigestValue>
      UWuYTYug10J1k5hKfonxthgrAR8=
     </dsig:DigestValue>
    </dsig:Reference>
  </dsig:SignedInfo>
```

```
    <dsig:SignatureValue>
     IWijxQjUrcXBYoCei4QxjWo9Kg8D3p9tlWoT4t0/gyTE96639In0FZFY2/rvP+/
     bMJ01EArmKZsR5VW3rwoPxw=
    </dsig:SignatureValue>
    <dsig:KeyInfo>
      <dsig:KeyValue>
         <dsig:RSAKeyValue>
           <dsig:Modulus>
            xA7SEU+e0yQH5rm9kbCDN9o3aPIo7HbP7tX6WOocLZAtNfyxSZDU16ks
            L6WjubafOqNEpcwR3RdFsT7bCqnXPBe5ELh5u4VEy19MzxkXRgrMvavz
            yBpVRgBUwUlV5foK5hhmbktQhyNdy/6LpQRhDUDsTvK+g9Ucj47es9AQ
            J3U=
           </dsig:Modulus>
           <dsig:Exponent>AQAB</Exponent>
         </dsig:RSAKeyValue>
      </dsig:KeyValue>
    </dsig:KeyInfo>
  </dsig:Signature>
```

Annex 1: Exemplary XML data sets in full

Disclaimer: This is a completely fabricated, hypothetical data set for demonstration purposes only. All content is for demonstration purposes and names of content titles or organisations are only used for practical clarification.

Example of shortest possible version of a MIRACLE data set in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<label xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
 xmlns="http://www.miracle-label.eu/ns/2.0/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.miracle-label.eu/ns/2.0/
 http://www.miracle-label.eu/ns/2.0/miracle-2-0.xsd">
  <age-declaration>
    <issuer>
       <age-issuer>myhomepage.cz</age-issuer>
    </issuer>
    <scope>
       <scope-urls>
          <scope-url>*.myhomepage.cz/*</scope-url>
       <scope-urls>
    </scope>
    <rating>
        <age>12</age>
    </rating>
  </age-declaration>
</label>
```

Long version with different examples

(Note: Some information might not make sense in practice, as this example is for documentation purposes only)

```
<?xml version="1.0" encoding="UTF-8"?>
<label xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
 xmlns="http://www.miracle-label.eu/ns/2.0/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.miracle-label.eu/ns/2.0/
http://www.miracle-label.eu/ns/2.0/miracle-2-0.xsd">
  <age-declaration>
    <issuer>
        <age-issuer>PEGI</age-issuer>
        <issuer-url>http://www.pegi.eu</issuer-url>
        <issuer-licence>C0-2124-23443</issuer-licence>
        <last-change>2014-06-25</last-change>
        <country>
           <country-code>BE</country-code>
           <country-code>DK</country-code>
           <country-code>FI</country-code>
           <country-code>FR</country-code>
           <country-code>GR</country-code>
           <country-code>AT</country-code>
           <country-code>IE</country-code>
           <country-code>IT</country-code>
```

```
          <country-code>LU</country-code>
          <country-code>NL</country-code>
          <country-code>NO</country-code>
          <country-code>PT</country-code>
          <country-code>ES</country-code>
          <country-code>SE</country-code>
          <country-code>CH</country-code>
          <country-code>GB</country-code>
          <country-code>EE</country-code>
          <country-code>CZ</country-code>
          <country-code>HU</country-code>
          <country-code>LT</country-code>
          <country-code>LV</country-code>
          <country-code>PL</country-code>
          <country-code>SK</country-code>
          <country-code>SI</country-code>
          <country-code>IS</country-code>
          <country-code>MT</country-code>
          <country-code>CY</country-code>
          <country-code>RO</country-code>
          <country-code>BG</country-code>
          <country-code>IL</country-code>
        </country>
        <custom>
            <custom-field class="PEGI-custom">
            PEGI-related custom field content
            </custom-field>
        </custom>
    </issuer>
    <scope>
      <scope-urls>
        <scope-url class="web-url">*.example.com/supergame/*
        </scope-url>
      </scope-urls>
      <scope-ids>
        <scope-id class="PEGI-classification-no">18423</scope-id>
      </scope-ids>
      <scope-titles>
        <scope-title class="PEGI-title-en" language="en">Supergame
Title
        </scope-title>
         <scope-title class="PEGI-title-de" language="de">Superspiel
Titel
        </scope-title>
        <scope-title class="PEGI-version">1.2</scope-title>
        <scope-title class="PEGI-language">en</scope-title>
      </scope-titles>
    </scope>
    <rating>
        <age>12</age>
      <age-adds>
        <age-add class="PEGI-age">pegi12+</age-add>
      </age-adds>
      <age-icons>
        <age-icon class="PEGI-icon">
        http://pegi.eu/label/12.png</age-icon>
      </age-icons>
```

```
    </rating>
    <content-descriptors>
        <cd-violence>
            <cd-violence-exist>true</cd-violence-exist>
            <cd-violence-icon>http://pegi.eu/label/violence.png
            </cd-violence-icon>
        </cd-violence>
        <cd-fear>
            <cd-fear-exist>true</cd-fear-exist>
            <cd-fear-icon>http://pegi.eu/label/fear.png
            </cd-fear-icon>
        </cd-fear>
        <cd-other>
            <cd-add class="self-harm">
                <cd-add-exist>true</cd-add-exist>
                <cd-add-icon>http://pegi.eu/label/self-harm.png
                </cd-add-icon>
            </cd-add>
        </cd-other>
        <cd-opentext class="consumerAdvice">
            <cd-opentext-desc> The game was rated PEGI 18 for scenes
            of sexual violence towards human characters.
            Not appropriate for persons below 18 years of age.
            </cd-opentext-desc>
        </cd-opentext>
    </content-descriptors>
    <feature-descriptors>
        <fd-inapppurchase>
            <fd-inapppurchase-exist>true</fd-inapppurchase-exist>
            <fd-inapppurchase-icon>http://pegi.eu/label/iap.png
            </fd-inapppurchase-icon>
        </fd-inapppurchase>
        <fd-personaldatasharing>
            <fd-personaldatasharing-exist>true
            </fd-personaldatasharing-exist>
            <fd-personaldatasharing-icon>
            http://pegi.eu/label/pds.png
            </fd-personaldatasharing-icon>
        </fd-personaldatasharing>
        <fd-chat>
            <fd-chat-exist>true</fd-chat-exist>
            <fd-chat-icon>http://pegi.eu/label/chat.png
            </fd-chat-icon>
        </fd-chat>
           <fd-other>
            <fd-add class="self-harm">
                <fd-add-exist>true</fd-add-exist>
                <fd-add-icon>http://pegi.eu/label/self-harm.png
                </fd-add-icon>
            </fd-add>
        </fd-other>
           <fd-opentext>
               <fd-opentext-desc class="PEGI-fd-info">
               Text regarding additional features</fd-opentext-desc>
           </fd-opentext>
    </feature-descriptors>
  </age-declaration>
</label>
```

Annex 2: XSD for MIRACLE XML data set

The XSD file and its technical documentation is available at
http://www.miracle-label.eu/data-model/

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.miracle-label.eu/ns/2.0/"
xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
    version="2.0" targetNamespace="http://www.miracle-label.eu/ns/2.0/" elementFormDefault="qualified"
    vc:maxVersion="1.1" vc:minVersion="1.0" xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning">

    <!-- Importing W3C xmldsig-core -->

    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd"/>

    <!-- Inclusion of ISO country code XSD -->
    <xs:include schemaLocation="http://www.miracle-label.eu/ns/ref/isocountrycodes.xsd"/>
    <!-- Inclusion of ISO country code XSD -->
    <xs:include schemaLocation="http://www.miracle-label.eu/ns/ref/isolangcodes.xsd"/>


    <!-- Root element of classification dataset(s) -->

    <xs:element name="label" type="label"/>

    <xs:complexType name="label">
        <xs:sequence minOccurs="1" maxOccurs="unbounded">
            <xs:element name="age-declaration" type="age-declaration"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Main building blocks -->
    <xs:complexType name="age-declaration">
        <xs:all>
            <xs:element name="issuer" type="issuer" maxOccurs="1" minOccurs="1"/>
            <xs:element name="scope" type="scope" maxOccurs="1" minOccurs="0"/>
            <xs:element name="rating" type="rating" maxOccurs="1" minOccurs="1"/>
            <xs:element name="content-descriptors" type="content-descriptors" maxOccurs="1"
                minOccurs="0"/>
            <xs:element name="feature-descriptors" type="feature-descriptors" maxOccurs="1"
                minOccurs="0"/>
            <xs:element ref="dsig:Signature" maxOccurs="1" minOccurs="0"/>
        </xs:all>
        <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
    <!-- End of main building blocks -->


    <!-- First block: information regarding the issuer -->
    <xs:complexType name="issuer">
        <xs:all>
            <xs:element name="age-issuer" type="xs:string" maxOccurs="1" minOccurs="1"/>
            <xs:element name="issuer-url" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
            <xs:element name="issuer-licence" type="xs:anyURI" minOccurs="0"/>
            <xs:element name="last-change" type="xs:date" maxOccurs="1" minOccurs="0"/>
            <xs:element name="country" type="country" maxOccurs="1" minOccurs="0"/>
            <xs:element name="customer-licence" type="xs:string" minOccurs="0"/>
            <xs:element name="custom" type="custom" maxOccurs="1" minOccurs="0"/>
        </xs:all>
    </xs:complexType>
    <!-- End of first block -->

    <!-- Second block: information regarding the scope -->
    <xs:complexType name="scope">
        <xs:all>
            <xs:element name="scope-urls" type="scope-urls" maxOccurs="1" minOccurs="0"/>
            <xs:element name="scope-ids" type="scope-ids" maxOccurs="1" minOccurs="0"/>
            <xs:element name="scope-titles" type="scope-titles" maxOccurs="1" minOccurs="0"/>
            <xs:element name="scope-hashes" type="hashTypes" maxOccurs="1" minOccurs="0"/>
                            <xs:element name="scope-api" type="xs:anyURI" maxOccurs="1" minOccurs="0"/>
                            <xs:element name="scope-metadata" type="scope-metadata" maxOccurs="1" minOccurs="0"/>
        </xs:all>
    </xs:complexType>
    <!-- End of second block -->

    <!-- Third block: information regarding the age rating -->
    <xs:complexType name="rating">
        <xs:sequence>
            <xs:element name="age" type="age" maxOccurs="1" minOccurs="1"/>
            <xs:element name="age-adds" type="age-adds" maxOccurs="1" minOccurs="0"/>
            <xs:element name="age-icons" type="age-icons" maxOccurs="1" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <!-- End of third block -->

    <!-- Fourth block: information regarding the content descriptors -->
    <xs:complexType name="content-descriptors">
        <xs:all>
            <xs:element name="cd-sexuality" type="cd-sexuality" minOccurs="0" maxOccurs="1"/>
            <xs:element name="cd-violence" type="cd-violence" minOccurs="0" maxOccurs="1"/>
            <xs:element name="cd-discrimination" type="cd-discrimination" minOccurs="0"
                maxOccurs="1"/>
            <xs:element name="cd-cursing" type="cd-cursing" minOccurs="0" maxOccurs="1"/>
            <xs:element name="cd-drugs" type="cd-drugs" minOccurs="0" maxOccurs="1"/>
```

```
        <xs:element name="cd-fear" type="cd-fear" minOccurs="0" maxOccurs="1"/>
        <xs:element name="cd-gambling" type="cd-gambling" minOccurs="0" maxOccurs="1"/>
        <xs:element name="cd-other" type="cd-other" minOccurs="0" maxOccurs="1"/>
        <xs:element name="cd-opentext" type="cd-opentext" minOccurs="0" maxOccurs="1"/>
    </xs:all>
</xs:complexType>
<!-- End of fourth block -->

<!-- Fifth block: information regarding the feature descriptors -->
<xs:complexType name="feature-descriptors">
    <xs:all>
        <xs:element name="fd-inapppurchase" type="fd-inapppurchase" minOccurs="0" maxOccurs="1"/>
        <xs:element name="fd-personaldatasharing" type="fd-personaldatasharing" minOccurs="0"
            maxOccurs="1"/>
        <xs:element name="fd-locationdatasharing" type="fd-locationdatasharing" minOccurs="0"
            maxOccurs="1"/>
        <xs:element name="fd-chat" type="fd-chat" minOccurs="0" maxOccurs="1"/>
        <xs:element name="fd-other" type="fd-other" minOccurs="0" maxOccurs="1"/>
        <xs:element name="fd-opentext" type="fd-opentext" minOccurs="0" maxOccurs="1"/>
    </xs:all>
</xs:complexType>
<!-- End of fifth block -->

<!-- ######################################### -->
<!-- Single type definitions below this line -->
<!-- ######################################### -->

<!-- Type definitions - first block: issuer -->

<xs:complexType name="country">
    <xs:sequence>
        <xs:element name="country-code" maxOccurs="unbounded" minOccurs="0" type="isocountrycode"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="custom">
    <xs:sequence>
        <xs:element name="custom-field" type="custom-field" maxOccurs="unbounded" minOccurs="0"
        />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="custom-field">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>


<!-- Type definitions - second block: scope -->

<xs:complexType name="scope-urls">
    <xs:sequence>
        <xs:element name="scope-url" type="scope-url" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="scope-url">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="scope-ids">
    <xs:sequence>
        <xs:element name="scope-id" type="scope-id" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="scope-id">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="scope-titles">
    <xs:sequence>
        <xs:element name="scope-title" type="scope-title" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="scope-title">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
            <xs:attribute name="language"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:attributeGroup name="language">
    <xs:attribute ref="lang">
        <xs:annotation>
```

```
            <xs:documentation>Reference to two-letter ISO language code schema.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
</xs:attributeGroup>

<xs:complexType name="hashTypes">
    <xs:sequence>
        <xs:element name="hashType" type="hashType" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:simpleType name="hashType">
    <xs:restriction base="xs:string">
        <xs:length value="32"/>
        <xs:pattern value="([a-z]|[0-9])*"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="scope-metadata">
    <xs:sequence>
        <xs:element name="scope-meta" type="scope-meta" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="scope-meta">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<!-- Type definitions - third block: rating -->

<xs:simpleType name="age">
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="-1"/>
        <xs:maxInclusive value="99"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="age-adds">
    <xs:sequence>
        <xs:element name="age-add" type="age-add" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="age-add">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:complexType name="age-icons">
    <xs:sequence>
        <xs:element name="age-icon" type="age-icon" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="age-icon">
    <xs:simpleContent>
        <xs:extension base="xs:anyURI">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>


<!-- Type definitions - fourth block: content-descriptors -->

<xs:complexType name="cd-sexuality">
    <xs:all>
        <xs:element name="cd-sexuality-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="cd-sexuality-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-sexuality-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-violence">
    <xs:all>
        <xs:element name="cd-violence-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="cd-violence-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-violence-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-discrimination">
    <xs:all>
        <xs:element name="cd-discrimination-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="cd-discrimination-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-discrimination-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-cursing">
    <xs:all>
        <xs:element name="cd-cursing-exist" type="xs:boolean" minOccurs="0"/>
```

```
        <xs:element name="cd-cursing-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-cursing-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-drugs">
    <xs:all>
        <xs:element name="cd-drugs-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="cd-drugs-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-drugs-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-fear">
    <xs:all>
        <xs:element name="cd-fear-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="cd-fear-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-fear-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-gambling">
    <xs:all>
        <xs:element name="cd-gambling-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="cd-gambling-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="cd-gambling-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="cd-other">
    <xs:sequence>
        <xs:element name="cd-add" type="cd-add" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="cd-add">
    <xs:all>
        <xs:element name="cd-add-exist" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
        <xs:element name="cd-add-desc" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="cd-add-icon" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
    </xs:all>
    <xs:attribute name="class" type="xs:string"/>
</xs:complexType>

<xs:complexType name="cd-opentext">
    <xs:sequence>
        <xs:element name="cd-opentext-desc" type="cd-opentext-desc" minOccurs="0"
            maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="cd-opentext-desc">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="class" type="xs:string"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<!-- Type definitions - fifth block: feature-descriptors -->

<xs:complexType name="fd-inapppurchase">
    <xs:all>
        <xs:element name="fd-inapppurchase-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="fd-inapppurchase-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="fd-inapppurchase-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="fd-personaldatasharing">
    <xs:all>
        <xs:element name="fd-personaldatasharing-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="fd-personaldatasharing-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="fd-personaldatasharing-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="fd-locationdatasharing">
    <xs:all>
        <xs:element name="fd-locationdatasharing-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="fd-locationdatasharing-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="fd-locationdatasharing-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="fd-chat">
    <xs:all>
        <xs:element name="fd-chat-exist" type="xs:boolean" minOccurs="0"/>
        <xs:element name="fd-chat-desc" type="xs:string" minOccurs="0"/>
        <xs:element name="fd-chat-icon" type="xs:anyURI" minOccurs="0"/>
    </xs:all>
</xs:complexType>

<xs:complexType name="fd-other">
    <xs:sequence>
        <xs:element name="fd-add" type="fd-add" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

```
    <xs:complexType name="fd-add">
        <xs:all>
            <xs:element name="fd-add-exist" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="fd-add-desc" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="fd-add-icon" type="xs:anyURI" minOccurs="0" maxOccurs="1"/>
        </xs:all>
        <xs:attribute name="class" type="xs:string"/>
    </xs:complexType>

    <xs:complexType name="fd-opentext">
        <xs:sequence>
            <xs:element name="fd-opentext-desc" type="fd-opentext-desc" minOccurs="0"
                maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="fd-opentext-desc">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="class" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```