Header information (Dublin Core Metadata Element Set)

The following examples contain a recommendation for the usage of Dublin Core (DC) Metadata Element Set of **<railML>.<metadata>** (at the root level of a RailML file). It is at the moment not necessary to use the DC elements in this way. But since it is not useful for an exchange of data if the same attribute has different meanings it is recommended to use the DC elements in the following way.

```
<railml version='2.0' … >
  <metadata>
    <dc:format>2.0.3</dc:format>
    <dc:identifier>1</dc:identifier>
    <dc:language>und-Latn</dc:language>
    <!--Codepage=1252  (ANSI - Lateinisch I)-->
    <dc:source>iPLAN.exe V1.2.0.528 NtzIntf_RailML2.dll V2.0.4.23</dc:source>
    <!--created with FBS (www.irfp.de) iPLAN.exe V1.2.0.528 NtzIntf_RailML2.dll V2.0.4.23-->
    <dc:source>Ostsachsen.ntz</dc:source>
    <dc:date>2012-03-01T11:14:59</dc:date>
    <dc:creator>iRFP</dc:creator>
  </metadata>
```

The attribute **railml.version** shall contain the „Marketing Version“ the scheme files were published with. Typical values are currently **2.0**, **2.1** and **2.2**. For pre-releases of future versions the release number of the schemes shall be given, e. g. **2.2r611** (for a pre-railML 2.3).

The **Dublin Core Metadata Element Set** (name space „dc:“, structure railml.metadata) is intended for implementation-depending version numbers and other header information.

The element **metadata.format** contains the internal version number of the scheme occurrence (also called RailML *profile*). This version number changes if the interpretation of the scheme by the writing software changes. A reading software should check whether this version number is equal or higher than the version number it was tested with. A reading software also can easily check with this number whether will be some special values which are
- necessary for the reading software,
- optional in RailML,
- obligatory in the specific scheme instance.

It is a matter of the writing software to assign the values for metadata.format. Therefore, these values can only be interpreted correctly together with metadata.source. Also it is a matter of the writing software to secure that there are only unique combinations of metadata.format and metadata.source (for identification of the writing software).

The element **metadata.identifier** contains a compatibility number as a simple integer value. This number only changes if the interpretation of an already existing value (attribute or element) changes after its first release (mostly following of an error correction). A reading software should check this value against a certain expected value - otherwise it risks that the data fields do no more contain the expected contents.

For an example, an attribute for speed values could contain speeds in km per hour. This confesses to be not conforming to RailML. To provide RailML conformity, this attribute would have to be changed to meters per second - without to be renamed. In this case metadata.identifier would be increased by one to avoid those reading programmes which do not know this change read it as kph.

The element metadata.identifier will not be changed if new data are added to a scheme. This is the main difference to metadata.format. It is expected that metadata.identifier changes very seldomly compared to metadata.format.

The element **metadata.source** has two functions:
a) To identify the writing software in a unique matter,

b) Optionally to name the data source (source database or source files).
This information is normally of less meaning for a reading programme but they may be used to identify the elements <metadata>.format and <metadata>.identifier.

The element **metadata.language** contains the BCP47 script tag of the character set (codepage) the data belongs to. This value is of importance in case the containing Unicode names (station names a.s.o.) have to be converted into a non-Unicode-string by the reading software.

This value is not to be mixed with `<?xml ... encoding='UTF-8' ?>` which defines the coding of the RailML file (and all of its characters). Since a RailML file is normally coded in UTF-8 the value of metadata.language is not necessary for pure reading. It is only necessary if the names have to be converted into a non-Unicode-string for the final target software. A reading algorithm shall not need to 'scan' the names for special characters which would mean a more empirical solution.

For example, in case of `<dc:language>und-Grek</dc:language>` a reading programme could - without this statement - only recognise that there are obviously Greek names by recognising high Unicode-(UTF-8-)values. If a writing programme does not know the origin of the names it shall skip the attribute metadata.language.

A problem is that there are no BCP47 tags for the European character sets of Latin script. There is the one and only *und-Latn* tag to be used for several codepages as Western Europe, Central Europe, Balic a. s. o. Therefore, a further naming of the codepage number such as

```
<!--Codepage=1252  (ANSI - Lateinisch I)-->
```

may be necessary but unfortunately there is no attribute for that.

The element **metadata.date** contains date and time of export (creation of the RailML file) in xs:dateTime format.

The element **metadata.creator** contains optionally the user name oder licence name of the user or licence used to create the file (e. g. login name of operating system or company name the writing software is licensed to).

Kopfinformationen (Dublin Core Metadata Element Set)

Die folgende Beschreibung enthält eine Empfehlung für die Verwendung des Dublin Core (DC) Metadata Element Set in **<railML>.<metadata>** (d. h. auf oberster Ebene der RailML-Datei). Momentan ist es nicht zwingend erforderlich, die DC-Elemente in der hier beschriebenen Art zu verwenden. Da es aber dem Datenaustausch nicht dienlich ist, gleiche Attribute mit verschiedenen Bedeutungen zu verwenden, wird die hier beschriebene Anwendung empfohlen. Vorkommen des Elements <metadata> in tieferen Ebenen (z. B. unter <railML>.<timetable>) beinhalten typischer Weise nur Teilmengen der Attribute (wie etwa *source*) und sollten sich ebenfalls an der hier dargestellten Verwendung orientieren.

```
<railml version='2.0' … >
  <metadata>
    <dc:format>2.0.3</dc:format>
    <dc:identifier>1</dc:identifier>
    <dc:language>und-Latn</dc:language>
    <!--Codepage=1252  (ANSI - Lateinisch I)-->
    <dc:source>iPLAN.exe V1.2.0.528 NtzIntf_RailML2.dll V2.0.4.23</dc:source>
    <!--created with FBS (www.irfp.de) iPLAN.exe V1.2.0.528 NtzIntf_RailML2.dll V2.0.4.23-->
    <dc:source>Ostsachsen.ntz</dc:source>
    <dc:date>2012-03-01T11:14:59</dc:date>
    <dc:creator>iRFP</dc:creator>
  </metadata>
```

Das Attribut **<railml>.version** soll die „Nenn-Version“ enthalten, unter der die RailML-Schemendateien veröffentlicht wurden. Typische Werte sind derzeit **2.0**, **2.1** und **2.2**. Bei Vorab-Veröffentlichungen oder Tests zukünftiger Versionen soll dort die Release-Nummer der RailML-Schemen angegeben sein, z. B. **2.2r611** (für RailML 2.3).

Implementierungsabhängige Versionsnummern und andere Kopfinformationen sind entsprechend RailML-Standard im **Dublin Core Metadata Element Set** (Namensraum „dc:“, Struktur <railml>.<metadata>) enthalten.

Das Attribut **<metadata>.format** enthält die interne Versionsnummer der Schemen-Ausprägung (auch RailML-Profil genannt). Diese Versionsnummer ändert sich dann, wenn sich die Interpretation oder die Vollständigkeit der Umsetzung des RailML-Schemas durch die schreibende Software ändert. Ein lesendes Programm sollte prüfen, dass diese Versionsnummer nicht niedriger ist als die Version, mit der das Programm frühestens getestet wurde. Insbesondere kann hiermit vom lesenden Programm einfach geprüft werden, ob bestimmte Daten vorhanden sein werden, die

- notwendig sind für das lesende Programm,
- in RailML als optional gekennzeichnet sind,
- in der konkreten Schemeninstanz obligatorisch sind.

Es ist Sache der Programmierung der schreibenden Software, die Werte für metadata.format festzulegen. Diese Werte sind daher i. d. R. nur im Zusammenhang mit metadata.source interpretierbar.

Das Attribut **<metadata>.identifier** soll eine Kompatibilitätsnummer als einfachen Integer-Wert enthalten. Diese Nummer wird nur dann geändert, wenn ein bestehender Datenwert (Attribut oder Element) nachträglich uminterpretiert wird (z. B. in Folge einer Fehlerkorrektur). Ein lesendes Programm sollte dieses Attribut auf exakt den erwarteten Wert prüfen – sonst läuft es Gefahr, dass die auszuwertenden Datenfelder nicht mehr den erwarteten Inhalt enthalten. Beispielsweise könnte ein Geschwindigkeitsfeld bis zu einem bestimmten Zeitpunkt die Geschwindigkeit in km/h enthalten. Es sei angenommen, dass sich das nachträglich als nicht RailML-konform herausstellte. Um die RailML-Konformität wiederherzustellen, müsste des Geschwindigkeitsfeld nachträglich – ohne umbenannt zu werden – auf die Einheit m/s umgestellt werden. In diesem Falle würde <metadata>.identifier um eins weitergezählt werden, um lesende, nicht aktualisierte Programme davon abzuhalten, den neuen Wert als km/h einzulesen.

Das Attribut <metadata>.identifier wird nicht aktualisiert, wenn neue Daten hinzukommen, was es von <metadata>.format unterscheidet. Es wird erwartet, dass sich <metadata>.identifier im Gegensatz zu metadata.format nur sehr selten ändert.

Das Attribut **<metadata>.source** dient zwei Funktionen, wofür mehrere Instanzen dieses Attributs vorkommen können:
a) der eindeutigen Identifizierung des schreibenden Programms,
b) der Benennung der Quelldaten (z. B. Datenbank, Dateien) sofern zutreffend.
Die Informationen selbst sind für lesende Programme wenig aussagekräftig, können aber zur Einordnung von <metadata>.format und <metadata>.identifier dienen.

Das Attribut **<metadata>.language** enthält das BCP47-script-Tag des Zeichensatzes, in dem die Daten im schreibenden Programm vorliegen. Dieser Wert kann z. B. von Bedeutung sein, wenn die in der RailML-Datei enthaltenen UTF-8-Zeichenketten (Bahnhofsnamen usw.) vom lesenden Programm in eine Nicht-Unicode-Zeichenkette umgewandelt werden müssen.

Dieser Wert ist nicht zu verwechseln mit `<?xml ... encoding='UTF-8' ?>`, welcher die Codierung der RailML-Datei und damit aller enthaltener Zeichen definiert. Da eine RailML-Datei normalerweise im Zeichensatz UTF-8 codiert ist, ist der Wert metadata.language zum reinen Auslesen der RailML-Datei nicht notwendig. Er ist nur von Bedeutung, wenn die Eigennamen aus der RailML-Datei im Zielprogramm in Nicht-Unicode-Zeichensätze umgewandelt werden müssen. Ein lesendes Programm soll hier nicht erst alle Eigennamen auf eventuell vorhandene Umlaute „scannen“ müssen, um sich auf eher empirische Weise für einen Zeichensatz zu entscheiden. Vielmehr soll das schreibende Programm – wannimmer möglich – quasi die Herkunft oder Zugehörigkeit der Namen zu einer Sprache angeben.

Beispielsweise wäre im Falle von `<dc:language>und-Grek</dc:language>` für ein lesendes Programm ohne diese Angabe erst durch Auffinden eigenartig hoher Unicode-(UTF-8-)Werte in den Eigennamen feststellbar, dass es sich offensichtlich um griechische Namen handelt. Sofern ein schreibendes Programm die geografische Zuordnung der Namen nicht feststellen kann, soll es metadata.language auslassen.

Problematisch ist allerdings, dass die BCP47-Tags nicht ausreichen, die verschiedenen europäischen Zeichensätze lateinischer Schrift zu benennen – es gibt praktisch nur das eine Tag *und-Latn*, welches für die Zeichnesätze (hier im Sinne von „Codepages“) Westeuropa, Mitteleuropa, Baltikum usw. verwendet werden muss. Aus diesem Grunde kann die zusätzliche Angabe der Codepage-Nummer wie etwa

```
<!--Codepage=1252  (ANSI - Lateinisch I)-->
```

notwendig werden, wofür derzeit aber leider kein spezielles Feld vorgesehen ist.

Das Attribut **<metadata>.date** enthält optional Datum und Uhrzeit des Exports (des Erzeugens der RailML-Datei) im xs:dateTime-Format.

Das Attribut **<metadata>.creator** kann optional den Anwender- oder Lizenznamen des Anwenders enthalten, der die Datei erzeugt hat (z. B. Anmeldename am Betriebssystem oder Firmenname, auf den die Programmlizenz ausgestellt ist.)