Uniform date formatting in XBRL inline XHTML reports

Kedves Loránd

Széchenyi István University, Artificial Intelligence National Laboratory (MILAB)

kedves.lorand@ga.sze.hu

Abstract

Since its introduction in 2003, XBRL became the global standard for business data reporting with wide support from various financial institutions, as well as the planned method of publishing and evaluating sustainability-related information. The key to its versatility is the clean separation of the core task (reporting facts in a precisely identified context) from the actual nature of these facts (measurements, aspects of the context) that are defined in external taxonomies. There is one notorious enemy of such standardization: dates. There are many different date formatting schemas that may even cause confusion by the order of the numbers representing the month and the day, let alone if only two digits represent the year. This article explains the scenario and the current efforts of embedding all national formats into the standard but details the technical complexity and possible inconvenience of this approach. As a conclusion, it recommends using one unambiguous format, which is already part of the XHTML transformation repository and is the primary date format in the XBRL environment.

XBRL – short introduction, report formats, and their roles [1], [2]

XBRL or eXtensible Business Reporting Language is a free and global framework of structured information exchange. The heart of an XBRL report is a list of *facts* that must have a numeric or non-numeric *value*, a *concept* identifier, and a *context* to which that fact belongs to. Numeric facts must also have *unit* and *precision* information. Contexts must contain an *entity* (generally the reporting organization), a *date instant or period*, and optionally, any further *classification* like a geographical region. The mentioned concepts, classification aspects and options are not part of the XBRL standard, they must be defined in external *taxonomies* provided by the relevant authorities (financial organizations, governments, etc.). XBRL only describes how such taxonomies should be defined and referred to from the reports or from each other.

The XBRL standard has a deep relationship to the XML (eXtensible Markup Language) format and ecosystem. An XBRL taxonomy definition uses standard XML tools: the concepts appear in XML Schema Definition (XSD) files, imported by using XLink and XML imports tags, the XML namespace mechanisms provide unique member references. This allows taxonomy creators to use existing, standard XML tools and only to learn the XBRL-related concepts that also appear in the same format. Similarly, the "native format" of an XBRL report is also XML (with the standard .xml or .xbrl extension), this allows report publishers to work in the same ecosystem as the taxonomy creators.

However, XBRL itself is an abstract standard and defines multiple formats to publish reports, each having different pros and cons as follows.

- XML (XBRL): content-oriented and optimized structure. This xml file first defines the contexts and units as separate identified tags, followed by the list of facts that refer to the contexts and units, thus minimizing the file size and making report management easier. The taxonomy creators, the report publishers and those who analyze them work with the same, standard XML tools; with proper XML background knowledge the whole environment is self-explanatory. However, this is also a compromise: extracting the data from source systems and transforming them into the report xml is complicated; and regardless of this effort, the result is not suitable for human consumption like a properly formatted business summary. The latter must be created from the XML or even from yet another intermediary data collection.
- JSON: data and process-oriented structure. The facts are independent JSON objects that contain all context and unit information. As there is no reference resolution, this format allows very simple fact processing, filtering on context data, and any fact subset remains self-contained. However, this results in larger file size, does not support context data modification because context data is embedded in all facts, and it requires different (JSON) tooling. It is unlikely that reporters would use JSON as their primary format; and even though report processors would benefit from it, they can produce it automatically from other formats.
- XHTML: presentation-oriented format. In this case, the report is an XML-compatible HTML file that any HTML editor tool can create, the XBRL compatibility is provided by extra inline XBRL tags. The report publisher uses these tags to create the required XBRL report environment: include taxonomies, define contexts and units. Then, wrap all XBRL facts of the XHTML report (numeric values and non-numeric fragments) and connect to the context and units using the proper tags. In this format, all facts are user-friendly text, including numbers and dates which must be converted back to data values, this conversion is controlled by the 'format' fact attribute. The XHTML format is arguably the hardest to produce or process, however, it is perfect for human consumption both printed and online. From the publishers' point of view, for a little extra effort (they execute the same data extraction and transformation, only to a slightly different format) they get one result that is suitable for both the authorities and the public. The report processors extract XBRL information from XML and XHTML. The latter is slightly more complicated, the files are significantly larger with the human-only content, but they can use the same tools and algorithms. The only critical difference is converting text values back to numeric or date data that appears only with this format and must be flawless.
- CSV: generation-optimized format. This consists of a JSON 'catalog' and multiple CSV files that contain data for the facts. A report publisher with a complex IT infrastructure and many source systems can automate the data extraction and conversion into the CSV files and create a JSON catalog to create a valid XBRL report. For a bit more initial investment, this environment allows automatic updates to the report data content while the structure or the target report taxonomy is the same, and they can follow changes by modifying the CSV generation scripts and the catalog. The report processors need a tool to import the CSV format but with that, it's just another XBRL input.

The XHTML is the only state-of-the-art presentation format: any web browser can display it. Furthermore, general XBRL browser plugins can extract the report structure and allow navigation in the report or display additional information from the referred taxonomies – without any extra effort from the publisher. Publishers would like to minimize the cost of making these reports, have experience with the HTML format and tools. With the planned extension of companies required to submit XBRL reports with their financial data and the new sustainability reports on the horizon, it can said that the vast majority of XBRL reports will use the XHTML format.

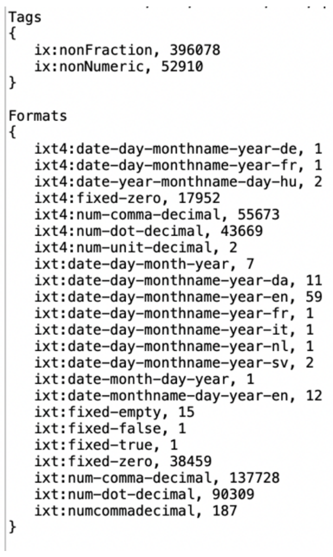
Consequently, it is very important to avoid any foreseeable problem related to the XHTML reports using the inline XBRL standard. Our research indicates that the current changes [3] in the transformation registry [4] may become such a problem by giving a technically challenging solution to a relatively minor problem.

Date facts in XHTML format

As mentioned above, each fact is related to one, exactly defined context, and that context must contain a date period or instant, like a 'bank deposit at a given time' or 'revenue over a given period'. Although the context always contains a date, there are facts of date type, like 'when the company transferred a monthly rent' or 'when was the Earth Overshoot Day this year'; but we assumed that there should be significantly less than numeric or text facts.

Date facts are rare in XBRL reports

This became obvious when our team started creating statistics over a random sample from the EU filings [5]. We processed 1000 randomly and 35 manually selected reports (from the available 7341, giving 14% of the total) and investigated the coverage of various formats, the raw log showed:



```
Tags
{
   ix:nonFraction, 396078
   ix:nonNumeric, 52910
}

Formats
{
   ixt4:date-day-monthname-year-de, 1
   ixt4:date-day-monthname-year-fr, 1
   ixt4:date-year-monthname-day-hu, 2
   ixt4:fixed-zero, 17952
   ixt4:num-comma-decimal, 55673
   ixt4:num-dot-decimal, 43669
   ixt4:num-unit-decimal, 2
   ixt:date-day-month-year, 7
   ixt:date-day-monthname-year-da, 11
   ixt:date-day-monthname-year-en, 59
   ixt:date-day-monthname-year-fr, 1
   ixt:date-day-monthname-year-it, 1
   ixt:date-day-monthname-year-nl, 1
   ixt:date-day-monthname-year-sv, 2
   ixt:date-month-day-year, 1
   ixt:date-monthname-day-year-en, 12
   ixt:fixed-empty, 15
   ixt:fixed-false, 1
   ixt:fixed-true, 1
   ixt:fixed-zero, 38459
   ixt:num-comma-decimal, 137728
   ixt:num-dot-decimal, 90309
   ixt:numcommadecimal, 187
}
```

To summarize this result, the 1035 reports contained 383979 numeric, 52811 text and 99 date facts. An XBRL report processor does not convert text facts, converts numeric values from text by learning if the decimal separator is comma or dot. However, for the 99 dates, there are 12 different formats in this sample, the registry contains multiple formats for each locale [4].

The occurrence ratio indicates that date facts are extremely rare, converting them from XHTML text to XBRL data should not be a complicated and error-prone activity.

Necessary text to date conversion in code

Unfortunately, this is not a “cosmetic task”: the reported fact is the text that a processor software must convert flawlessly into a date value. For example, these are the Czech variants in the transformation registry [4], (the blue block in each middle contains all allowed month names):

```xml
<xs:simpleType name="dateDayMonthnameCsType">
  <xs:annotation>
    <xs:documentation>
      Czech date in the order "day month". Accepts 1 or 2 digits for day.
      Accepts months in full or abbreviated form. Requires a non-numeric
      separator. The schema does not enforce valid day of month (e.g., it
      accepts "30. února").
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:pattern value="[0-9]{1,2}[^0-9]+(ledna|
    února|unora|března|brezna|dubna|května|kvetna|června|cervna|
    července|cervence|srpna|září|zari|října|rijna|listopadu|prosince|led|
    úno|uno|bře|bre|dub|kvě|kve|čvn|cvn|čvc|cvc|srp|zář|zar|
    říj|rij|lis|pro|LEDNA|ÚNORA|UNORA|BŘEZNA|BREZNA|DUBNA|KVĚTNA|KVETNA|
    ČERVNA|CERVNA|ČERVENCE|CERVENCE|SRPNA|ZÁŘÍ|ZARI|
    ŘÍJNA|RIJNA|LISTOPADU|PROSINCE|LED|ÚNO|UNO|BŘE|BRE|DUB|KVĚ|KVE|ČVN|CVN|
    ČVC|CVC|SRP|ZÁŘ|ZAR|ŘÍJ|RIJ|LIS|PRO|Ledna|
    Února|Unora|Března|Brezna|Dubna|Května|Kvetna|Června|Cervna|
    Července|Cervence|Srpna|Září|Zari|Října|Rijna|Listopadu|Prosince|Led|
    Úno|Uno|Bře|Bre|Dub|Kvě|Kve|Čvn|Cvn|Čvc|Cvc|Srp|Zář|Zar|
    Říj|Rij|Lis|Pro)\.?"/>
  </xs:restriction>
</xs:simpleType>
```

```xml
<xs:simpleType name="dateDayMonthnameYearCsType">
  <xs:annotation>
    <xs:documentation>
      Czech date in the order "day month year". Accepts 1 or 2 digits for day.
      Accepts months in full or abbreviated form. Accepts 1, 2 or 4 digits for
      year. Requires non-numeric separators. The schema does not enforce valid
      day of month (e.g., it accepts "30. února 2008").
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:pattern value="[0-9]{1,2}[^0-9]+(ledna|
    února|unora|března|brezna|dubna|května|kvetna|června|cervna|
    července|cervence|srpna|září|zari|října|rijna|listopadu|prosince|led|
    úno|uno|bře|bre|dub|kvě|kve|čvn|cvn|čvc|cvc|srp|zář|zar|
    říj|rij|lis|pro|LEDNA|ÚNORA|UNORA|BŘEZNA|BREZNA|DUBNA|KVĚTNA|KVETNA|
    ČERVNA|CERVNA|ČERVENCE|CERVENCE|SRPNA|ZÁŘÍ|ZARI|
    ŘÍJNA|RIJNA|LISTOPADU|PROSINCE|LED|ÚNO|UNO|BŘE|BRE|DUB|KVĚ|KVE|ČVN|CVN|
    ČVC|CVC|SRP|ZÁŘ|ZAR|ŘÍJ|RIJ|LIS|PRO|Ledna|
    Února|Unora|Března|Brezna|Dubna|Května|Kvetna|Června|Cervna|
    Července|Cervence|Srpna|Září|Zari|Října|Rijna|Listopadu|Prosince|Led|
    Úno|Uno|Bře|Bre|Dub|Kvě|Kve|Čvn|Cvn|Čvc|Cvc|Srp|Zář|Zar|Říj|Rij|Lis|Pro)[^0-
    9a-zA-Z]+[^0-9]*([0-9]{1,2}|[0-9]{4})"/>
  </xs:restriction>
</xs:simpleType>
```

```xml
<xs:simpleType name="dateMonthnameYearCsType">
  <xs:annotation>
    <xs:documentation>
      Czech date in the order "month year". Accepts months in full or
      abbreviated form. Accepts 1, 2 or 4 digits for year. Requires a non-
      numeric separator.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse"/>
    <xs:pattern value="(leden|ledna|lednu|únor|unor|února|unora|
    únoru|unoru|březen|brezen|března|brezna|březnu|breznu|duben|dubna|dubnu|květen|kveten|května|kvetna|květnu|kvetnu|
    červen|cerven|června|cervna|červnu|cervnu|červenec|cervenec|
    července|cervence|červenci|cervenci|srpen|srpna|srpnu|září|zari|říjen|rijen|
    října|rijna|říjnu|rijnu|listopad|listopadu|prosinec|prosince|prosinci|led|
    úno|uno|bře|bre|dub|kvě|kve|čvn|cvn|čvc|cvc|srp|zář|zar|
    říj|rij|lis|pro|LEDEN|LEDNA|LEDNU|ÚNOR|UNOR|ÚNORA|UNORA|
    ÚNORU|UNORU|BŘEZEN|BREZEN|BŘEZNA|BREZNA|BŘEZNU|BREZNU|DUBEN|DUBNA|DUBNU|KVĚTEN|KVETEN|KVĚTNA|KVETNA|KVĚTNU|KVETNU|
    ČERVEN|CERVEN|ČERVNA|CERVNA|ČERVNU|CERVNU|ČERVENEC|CERVENEC|
    ČERVENCE|CERVENCE|ČERVENCI|CERVENCI|SRPEN|SRPNA|SRPNU|ZÁŘÍ|ZARI|ŘÍJEN|RIJEN|
    ŘÍJNA|RIJNA|ŘÍJNU|RIJNU|LISTOPAD|LISTOPADU|PROSINEC|PROSINCE|PROSINCI|LED|
    ÚNO|UNO|BŘE|BRE|DUB|KVĚ|KVE|ČVN|CVN|ČVC|CVC|SRP|ZÁŘ|ZAR|
    ŘÍJ|RIJ|LIS|PRO|Leden|Ledna|Lednu|Únor|Unor|Února|Unora|
    Únoru|Unoru|Březen|Brezen|Března|Brezna|Březnu|Breznu|Duben|Dubna|Dubnu|Květen|Kveten|Května|Kvetna|Květnu|Kvetnu|
    Červen|Cerven|Června|Cervna|Červnu|Cervnu|Červenec|Cervenec|
    Července|Cervence|Červenci|Cervenci|Srpen|Srpna|Srpnu|Září|Zari|Říjen|Rijen|
    Října|Rijna|Říjnu|Rijnu|Listopad|Listopadu|Prosinec|Prosince|Prosinci|Led|
    Úno|Uno|Bře|Bre|Dub|Kvě|Kve|Čvn|Cvn|Čvc|Cvc|Srp|Zář|Zar|Říj|Rij|Lis|Pro)[^0-
    9a-zA-Z]+[^0-9]*([0-9]{1,2}|[0-9]{4})"/>
  </xs:restriction>
</xs:simpleType>
```

A programmer can imagine the code that must appear in all “conformant XHTML processor” software to convert all accepted month names in all templates back to a number in the 1-12 range, like a `Map<Integer, Set<String>>`, a `Map<String, Integer>`, a huge `switch(str) { }` or even an `if() { } else if() { }` … chain.

The problem here is that such codes (both production and unit tests!) are very sensitive to any typo and the error only pops up if that value appears in the data. That much effort and risk may not worth the custom formatting of that small percent of facts.

Date facts in inline XHTML reports from users’ perspective

A report publisher creates one report at a time and uses one locale in general, so following the previous example, would use one of the Czech date formats. But where do fact values come from? Either from a subsystem or entered by a user somewhere. If the value is generated by an export or selected by a date picker component, someone must use a function to create a text compatible with the selected formatting. Considering the wide range of options, an existing formatter would likely be compatible but should better be tested for all months. The only scenario when the above validation is relevant is when someone manually enters the date value as text.

However, multiple date formats can appear in a single report, like they show year in some dates but not in others. This also means that the original problem: the ambiguity caused by the order of numbers representing the day and the month is still not solved. Of course, additional validation codes can be created that exclude this scenario, but then this logic must appear in all XBRL report editors that create inline XHTML. Further complicating the issue, the selected format option is an xml attribute value, the web browser does not display it, so there is no indication on the screen about how to interpret the date value.

Of course, displaying the month name solves this problem and may be convenient for the report creator who is working in one “home” locale with official localization support. However, the whole reason for creating XBRL reports is to homogenize the knowledge over languages, regions. People will look at and probably try to compare these reports in their native form: in a web browser or printout. For them, it will not be convenient comparing date information when some of them appear with month names in various languages, while in others, they still see month and day numbers and may need web developer tools to reveal the value of the “format” xml attribute.

Our analysis does not show a real benefit for a user working on or reviewing a single report in its native environment, the web browser. For someone trying to work with multiple reports over the web or on paper, the localized date information will rather be a burden.

The one format to rule them all… is already available

To quote Antoine de Saint-Exupery, 'Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.' Although the goal here is not perfection, moving towards it is still better than to the opposite direction.

The essence of the XBRL ecosystem is to let authorities create taxonomies and expect report publishers to convert their local financial, sustainability, etc. information to that common terminology and structure to create a compliant report. When working with date information, data administrators and analysts tend to return to the ‘YYYY-MM-DD’ format because it is the global ISO standard, unambiguous, sortable etc. XBRL users are particularly familiar with it because this variant is required in all file formats to specify date for contexts, and except for the XHTML, even date facts.

Consequently, all existing data extractors generate date values for contexts and facts in YYYY-MM-DD format, their codes would need to be changed to use any other variant. The inline XBRL transformation registry already contains the `date-year-month-day` format, the only one that does not require any transformation because the source text is already in the target format. This is the safest and simplest, already supported solution, XHTML can and should use it as is.

Summary

Our core research focuses on creating a flexible management platform to handle XBRL reports. Investigating the specialties of each XBRL file format led to the conclusion that the XHTML with inline XBRL likely is and will be the most popular one. XHTML input analysis showed that date facts are exceptionally rare, proven by a 10% random sample from the current EU filings. The newer version of the XHTML transformation registry aims to incorporate all national date formats with written month names. This requires complex and error-prone processor software while the benefit of such extension is questionable. As a conclusion, we recommend using the ISO 8601 compatible YYYY-MM-DD format for date facts in the XHTML inline XBRL files. That is already used to provide dates in all other XBRL files (including XHTML for context dates), exists in the XHTML transformation registry as `date-year-month-day` since its initial version, therefore, the existing XBRL toolchain can work with it.

References

1. *XBRL Specifications* https://specifications.xbrl.org/specifications.html
2. Ghislain Fourny, *The XBRL Book: Simple, Precise, Technical* sixth edition (2023).
3. *Inline XBRL Transformation Registry v4 Presentation* (2019) https://www.xbrleurope.org/wp-content/uploads/2019/02/tr4.pdf
4. *Inline XBRL Transformation Registry v4 Specification* (2020) https://www.xbrl.org/Specification/inlineXBRL-transformationRegistry/REC-2020-02-12/inlineXBRL-transformationRegistry-REC-2020-02-12.html
5. Collection of EU XBRL reports from most Officially Appointed Mechanism (OAM) (online service) https://filings.xbrl.org/