X-definition 4.2

XML, JSON, YAML, XON

XML Prague 2022

Václav Trojan *trojan@syntea.cz*

Tomáš Šmíd *tomas.smid@syntea.cz*

syntea

What is X-definition

- X-definition is a programming language that was originally designed to work with XML (presented at XML Prague 2005). It is opensource project available at https://github.com/syntea/xdef.
- Each X-definition is written as an XML document.
- One X-definition can describe the structure of one or more models of objects and declaration of methods, types, and variables. X-definition may contain reference to another X-definition. Collection of source data with X-definitions is compiled to a Java class.
- The X-definition starting with the version 4.1 describes data models in JSON, YAML, Properties, Windows INI, or CSV
- Processed data can be converted to the special format XON.

What is XML and JSON

XML

The XML data model in Java is a w3c.dom.Document. Values of attributes or of text nodes are strings.

Actual work with XML data requires knowledge of the properties of the values that are described in the XML Schema, the RELAX-NG document, or the X-definition.

JSON

The JSON data model in Java is either java.util Map or java.util.List or String, Number, Boolean or null.

The advantage of JSON objects over the W3C DOM is that values are not just strings, but also numbers, boolean values, or null.

The properties of the JSON data object can be described externally in the JSON schema or in the X-definition.

syntea

XML -> JSON

Any XML object can be easily converted to JSON structure:

- A JSON map with one named value, which is a JSON array.
- The first item in this field is a JSON map containing attributes as named values (if attributes are missing, it can be omitted).
- The following array items are child nodes, that is, elements converted to JSON map, or JSON text values.

XML -> JSON example

XML:

```
<Book
    Title = "X-definition 4.2"
    Date = "2022-06" >
   <Author>
      Václav Trojan
   </Author>
   <Author>
      Tomáš Šmíd
   </Author>
</Book>
```

JSON:

```
{ "Book" :
  [
   { Title : "X-definition 4.2",
      Date : "2022-06" },
   {"Author" :
     ["Tomáš Šmíd"] },
   {"Author" :
     ["Václav Trojan"] }
  ]
}
```

X-definition model of XML element:

In X-definition model, values of attributes and text nodes can be processed by validation methods. It is possible to determine the occurrence of an item:

```
<xd:def xmlns:xd = "http://www.xdef.org/xdef/4.2"
        xd:root = "Book" xd:name = "Book" >
  <Book
     Title = "required string(1,120);"
     Date  = "optional gYearMonth();" >

       <Author xd:script = "occurs 0..*" >
           required string();
       </Author>
  </Book>
</xd:def>
```

X-definition model of JSON data

In JSON model, primitive values can be described as in XML model. However, the model is written using special element <xd:xon>. The occurrence of a values can be determined similarly: (see X-definition - JSON, XON, YAML (syntea.cz))

```
<xd:def xmlns:xd = "http://www.xdef.org/xdef/4.2"
        xd:root = "Book" xd:name = "Book" >
<xd:xon name="book">
{ "Book" : [
     { "Title" : "required string();",
       "Date"  : "optional gYearMonth();" },
     [ { x:script = "occurs 0..*;",
         "Author" : "required string();"
       }
     ]
  ]
}
</xd:xon>
</xd:def>
```

syntea

JSON -> XML

Converting JSON to XML is more complicated. JSON cannot be converted easily to XML because:

- JSON can be a map, an array or a primitive value.
- The map can be empty, contain one or more items – a map, an array or a primitive value.
- Name of a JSON map item can be any string, even an empty string (not an XML name).
- The array can be empty, contain one or more items – a map, an array or a primitive value.

syntea

JSON -> XML

If the structure of the JSON object matches the structure of the XML converted to JSON, the conversion is fairly easy, i.e. it must have the following properties:

- JSON map with one named value of the array type.
- The first item of the array is a map of strings (may be missing if the map is empty).
- The following items are either strings or maps that have a single named value that is an array with the properties specified above.

Problem of JSON arrays and maps

If the JSON structure of the map or JSON field does not match the structure described in the previous slide, it is converted to an XML element with the namespace URI "http://www.xdef.org/xon/4.0" (we use here the prefix "jx"):

- <jx:map> for map.
- <jx:array> for array.
- Number, Boolean value and null is converted as a string
- JSON string is embedded in quotes
- Array primitive values are converted to the element <xd:item> with the value in attribute "val".

Problem of JSON names and XML names

- JSON empty string name is converted to "_x_".
- Characters of a JSON name which would form invalid XML name are replaced with the sequence "_x*hh*_" (where „*hh*“ is an UTF-16 hexadecimal representation of the character without leading zeroes).
- If JSON name is "_x_" it is converted to ”_x5f_x_“.
- If JSON name contains character sequence "_x*hh*_" it is converted to "_x5f_x*hh*_" (where hh is a sequence of hexadecimal digits).

This algorithm guarantees the full conversion of JSON name to XML name and vice versa.

JSON -> XML example

JSON:

```
{ "Cities" : [
    "2020-02-22",
    { "from":
      [
        "Brussels",
        { "to": "London",
          "distance": 322
        },
        { "to": "Paris",
          "distance": 265
        }
      ]
    }
]}
```

XML:

```
<Cities>
  2020-02-22
  <from>
    Brussels
    <jx:map xmlns:jx = https://www.xdef.xon/4.0">
      <to val = "London" />
      <distance val = “322" />
    </jx:map>
    <jx:map xmlns:jx = "https://www.xdef.xon/4.0">
      <to val = "Paris" />
      <distance val = "265" />
    </jx:map>
  </from>
</Cities>
```

XON properties

- The XON format implements the basic X-definition types
- When a document is processed using X-definition, it is possible to get an XON object as a result.
- The XON object can be used as a general format for transferring data between systems.
- XON object can be used for unique form of XML, JSON, etc.

XON – X-definition Object Notation

The XON data model is an extension of the JSON model. However, it allows a more detailed description of primitive values:

- Number: byte, short, int, long, float, double, decimal, integer (BigInteger)
- Date and time (all types according to XML schema and X-definition: datetime, gYear etc.)
- Duration (all types according to XML schema and X-definition)
- byte array
- … (all types of values supported in X-definition)

JSON – sample

```
[
  {  a : 1,                          short
     b : "ab cd",                    string
     c : -INF,                       float negative infinity
     d : true,                       boolean
     e : "P1Y1M1DT1H1M1.12S",        Duration
     f : "HbRBHbRBHQw=",             Base64 (byte array)
     g : "2021-06-02",               Date
     h : "x",                        Character
     "sale price" : "12.50 CZK", Price
     Towns : [
          "48.2, 16.37, 151, Wien",          GPS
          "50.08, 14.42, 399, Praha-centrum" GPS
     ]
  }
]
```

XON – sample

```
%charset="ISO8859-2"          # charset of source data
[
  {  a : 1s,                  /* Short */
     b : "ab cd",             /* String */
     c : -INFf,               /* Float negative infinity */
     d : true,                /* Boolean */
     e : P1Y1M1DT1H1M1.12S,   /* Duration */
     f : b(HbRBHbRBHQw=),     /* byte array */
     g : D2021-06-02,         /* Date */
     h : c"x",                /* char */
     "sale price" : p(12.50 CZK),   /* Price */
     Towns : [ # array with GPS locations of towns
          g(48.2, 16.37, 151, Wien),
          g(50.08, 14.42, 399, Praha-centrum),
     ]
  }
]
```

YAML – Yet Another Markup Language

YAML is an alternative JSON notation. It contains the same types of values. JSON syntax is a subset of YAML. The YAML data model is the same as JSON:

- Map - Java Map, containing value names and their corresponding values
- Array - Java Array, array elements are JSON values
- Primitive value - Java Number, String, Boolean or null.

Properties, INI, CSV

Properties are used as configuration files for various applications. Data model is:

- Java Map containing the names of the corresponding string values

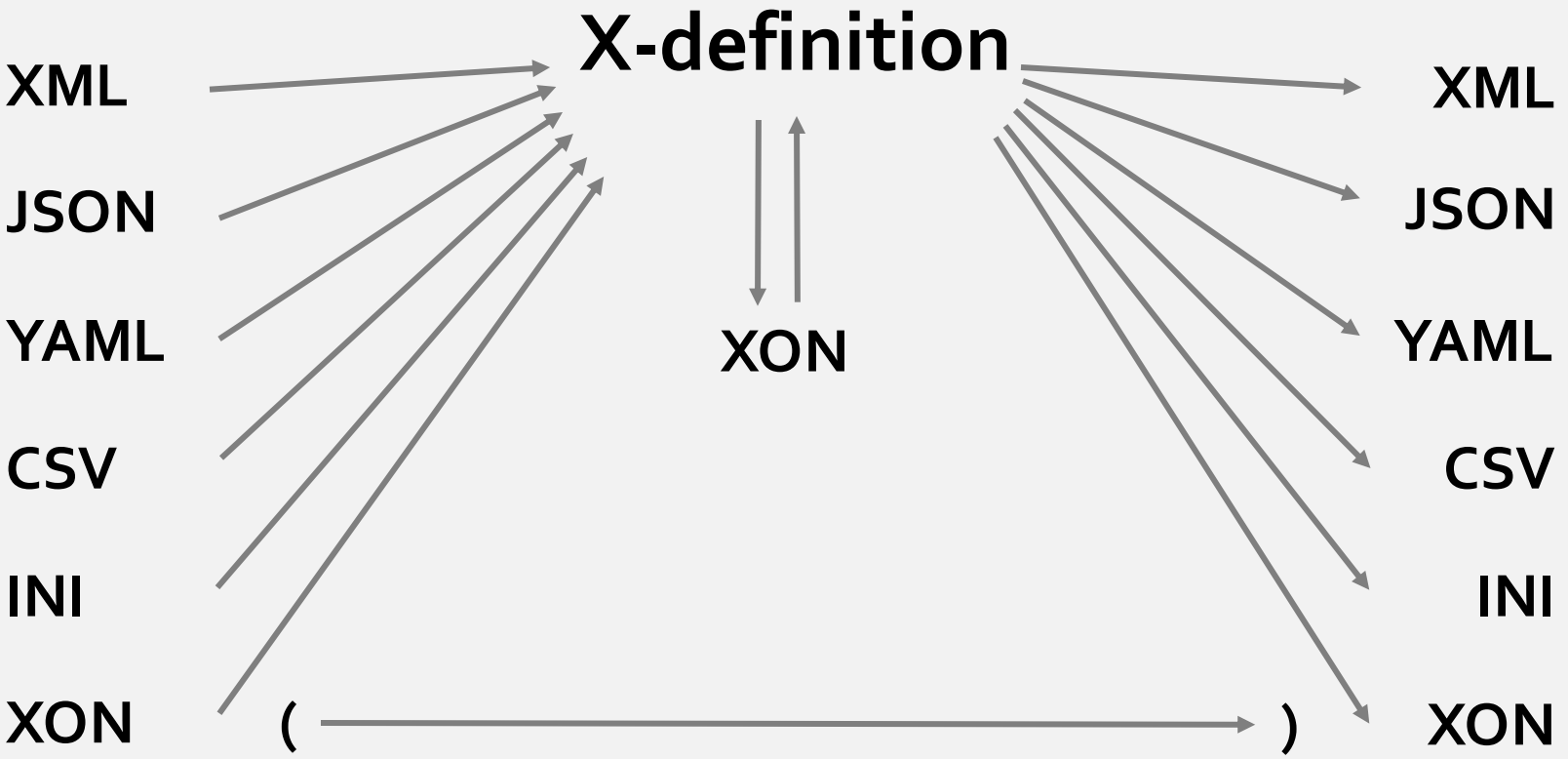
INI files are used as configuration files for various applications. They contain a map of named string values or a list of named sets of named strings. Data model:

- Java Map containing the names of the corresponding values, the value being a string or a nested Map.

CSV files are used mostly as spreadsheet data. They contain a sequence of lines where each line contains an array of primitive values separated by a comma (or other character). Data model:

- Java List containing sequence of List objects created from each line.

X-definition and XON



EXAMPLES OF XON PRACTICAL USE

syntea

Examples of XON practical use

JSON Input data

```
{ "products": [
   { "productId": 123456,
     "productName": "A green door",
     "price": 12.50,
     "tags": [ "home", "green" ]
   },
   { "productId": 987654,
     "productName": "bicycle",
     "price": 320.00,
     "tags": [ "bicycle", "silver", "electric" ]
   }
  ]
}
```

syntea

Examples of XON practical use

XON Input data

```
{ "products": [
   { "productId": 123456i,
     "productName": "A green door",
     "price": 12.50D,
     "tags": [ "home", "green" ]
   },
   { "productId": 987654i,
     "productName": "bicycle",
     "price": 320.00D,
     "tags": [ "bicycle", "silver", "electric" ]
   }
  ]
}
```

Examples of XON practical use

Data model definition

```
<xd:def xmlns:xd = "http://www.xdef.org/xdef/4.1" name = "Example" root = "product">
  <xd:xon xd:name = "product">
  { products:
    [
      {x:script="occurs *",
         "productId": "int(100000,999999)",
         "productName" : "string()",
         "price": "decimal()"
         "tags": ["occurs 0..* string()"]
      }
    ]
  }
  </xd:xon>
</xd:def>
```

syntea

Examples of XON practical use

- Same data validation model for JSON and XON
- Support of **stream data processing**
  - Ability to process large data files (size of GBs)
  - XML, JSON, XON, …

Examples of XON practical use

Telecommunication use case

- Exchange of large data files containing network device information
- Data cannot be processed in a basic way – loading the whole file into memory
  - *Note: Most of the standard tools do not support incremental parsing*
    - Need to program your own data analysis
    - Focus on the technical aspect instead of business goals
    - … or we can use X-definition (**rescue**)

Examples of XON practical use

Telecommunication use case: one record of JSON payload


```
{
  "container":false,
  "res-id":"fda88901-3b7f-33ec-80f260a95523fd7c",
  "roles":["MXU"],
  "communication-state":"1",
  "remark":"",
  "is-gateway":-1,
  "ip-address":"172.25.39.18",
  "admin-status":"inactive",
  "location":"",
  "physical-id":1
}
```

Examples of XON practical use

Telecommunication use case: X-definition

```
<xd:declaration>
  type id = string(%length=36);
</xd:declaration>

<xd:xon name='network'>
{
  "container":            "boolean()",
  "res-id":               "string(id)",
  "roles": [
    "occurs 1..* enum('OLT','MXU', 'OTHER');"
  ],
  ...
```

Examples of XON practical use

Telecommunication use case: X-definition

```
  "mac":                  "optional id()",
  "ref-parent-subnet":    "optional id()",
  "dev-sys-name":         "optional string()",
  "communication-state":  "optional int(0,1)"
  "remark":               "optional string()",
  "is-gateway":           "optional int(-1,1)",
  "ip-address":           "optional ipAddr()",
  "admin-status":         "optional enum('active', 'inactive')",
  "location":             "optional string()",
  "physical-id":          "optional int(1, *)",
}
</xd:xon>
```

FUTURE DEVELOPMENT OF X-DEFINITION

syntea

Future development of X-definition

Current status

- Focus on accessibility
  - Open sourced in 2018
- Focus on platform independence
  - Impossibility to use without Java

Future development of X-definition

What's next?

- Create a **general standard**
- The standard should contain following definitions:
  - basic data types (string, int, double, date, …)
  - structure (elements, attributes, sequence, map, arrays, …)
  - events (*onTrue*, *onAbsence*, *finally*, …)
    - Triggered internally by data processing
  - external methods
    - e.g.: support for advanced validations, additional action based on events

Future development of X-definition

What's next?

- Data model validation transformation
  - Algorithm between X-definition and JSON Schema
  - Basic two-way transform algorithm for XML Schema already done

Thanks for your attention

X-definition on GitHub: https://github.com/Syntea/xdef

X-definition releases on Maven: https://search.maven.org/search?q=g:org.xdef

X-definition tutorial: https://xdef.syntea.cz/tutorial/