xcore – xsd content reporter

(Pre-release information, version 2023-08-21)

Change log

** 20230821*
Added call parameter `$dnamesExcluded`, a list of folders the content of which are excluded from input
** 20230802*
Added descriptions of installation and command-line interface
** 20230630*
(1) Prog change: Mandatory content items and their cardinality constraint in bold
(2) Prog change: Cardinality constraint of choice branches flagged by preceding "-"
(3) Prog change: Enumeration dictionaries integrated into Simple Type tables

The new generation of an OJP Schema Reporter is a generic, customisable XSD documentation tool.

The tool has a command-line interface and maps schema contents to a report. The report type is determined by call parameter `$report`. Currently, the following report types are supported:

- ***contab*** ("content tables") – an extended version of the "classic" report produced by the old generation tools `ojp/docs/ojp-*.xsl`
- ***edesc*** ("expanded component descriptors") – an XML representation of selected main schema components (top-level element declarations, type and group definitions), with group and base type references recursively expanded
- ***def*** ("component definitions") – copies of selected schema components, optionally with annotations removed
- ***desc*** ("component descriptors") – an XML representation of selected main schema components, in which references are not expanded

The tool can be used for creating reports of *all* major schema components (top-level element, type definitions, group definitions), or of a user-defined set determined by fine-grained name filters (e.g. all type definitions with a name starting with "vehicle", containing "parking" and ending with "structure").

This preliminary documentation is restricted to the *contab* report, meant to replace the output produced by old generation tools. Some brief remarks about the other report types follow.

The *edesc* report is an XML representation of *effective* component content, e.g. providing information about which parts of content are inherited from which base type. Note that the contab report is implemented by first generating an edesc report and transforming it into a contab report.

The *def* report amounts to an extraction of selected components. The possibility to strip annotation (parameter `$skipAnno`) greatly enhances the readability of component contents.

The *desc* report is primarily used for development purposes.

Installation

(1) Unzip the xcore archive, in a folder at your discretion.
(2) Install BaseX, version 10 or higher. If the operation system is Windows, download and execute the Windows installer, otherwise download and unzip the ZIP package.
(3) Edit file `$basex/bin/basex.bat` in order to increase available heap space. Replace line

```
set BASEX_JVM=-Xmx1200m %BASEX_JVM%
```

with

```
set BASEX_JVM=-Xmx4800m %BASEX_JVM%
```

Command-line interface

As the application is written in XQuery and meant to be processed by BaseX, it can be launched using the command-line interface of BaseX.

Usage - general pattern

Note: whitespace is added to the call for readability, which must be removed. Optional options in square brackets.

```
basex  -b report=report-type
       -b dir=input-folder
       [-b odir=output-folder]
       [-b ofile=output-file-name]
        [-b dnamesExcluded=excluded-dir-names]
       [-s indent=yes]
       [-b custom=custom-file]
       [-b domains=domains-type]
       [-b edescReportDir=edesc-report-dir]
       [-b skipAnno=skip-anno-value]
       [-b enames=elem-name-pattern]
       [-b anames=att-name-pattern]
       [-b tnames=type-name-pattern]
       [-b gnames=group-name-pattern]
       [-b hnames=att-group-name-pattern]
       path/to/xcore/xcore.xq
```

where

- `report-type` = Report type (`contab` | `edesc` | `desc` | `def`)
- `input-folder` = Path to the folder containing the input XSDs (at any depth)
- `output-folder` = Path to the folder into which the output is written
- `output-file-name` = Parameter evaluated only if no domains have been defined: file name of the report file
- `excluded-dir-names` = Whitespace separated list of folder names; XSDs directly or indirectly contained by these folders are not considered as input; within folder names, replace whitespace characters with the string
- `indent=yes` = Parameter evaluated only if the report is not written into a file: indent XML data
- `custom-file` = Path to the optional customization file
- `domain-type` = If “xsd”, each XSD is described by a distinct report file
- `edesc-report-dir` = Parameter evaluated only if report=`contab`: path to the folder containing edesc reports `skip-anno-switch`
  = Parameter evaluated only if report=`def`: if =1, annotations are removed, otherwise retained
- `elem-name-pattern` = Ignore top-level elements not matching this name pattern
- `att-name-pattern` = Ignore top-level attributes not matching this name pattern
- `type-name-pattern` = Ignore type definitions not matching this name pattern
- `group-name-pattern` = Ignore group definitions not matching this name pattern
- `att-group-name-pattern` = Ignore att group definitions not matching this name pattern

Further explanations:

(1) *Relative paths* are resolved against the current working directory.

(2) *Output files* are written into a subfolder of `output-folder`; the name of the subfolder is equal to `report-type`

(3) If *domains* are defined, one report file is created for each domain. Otherwise, all output is written into a single file.

(4) If domains are defined, *parameter* `output-file-name` is ignored

(5) Domains can be ***defined*** in two different ways:
   - Using call parameter `domain-type=xsd` – one domain for each XSD
   - Using a custom file defining domains (element `domains`)

(6) In case of `domain-type=xsd`, the domain files are arranged in a *folder structure* mirroring the input folder structure; for example, an XSD located at

```
$input-folder/foo/bar/some.xsd
```

is mapped to a report file

```
$output-folder/report-name/foo/bar/some.xml
```

or

```
$output-folder/report-name/foo/bar/some.html
```

depending on the report type.

Usage examples – `contab` reports

The following examples assume:

1. `xcore` is installed as a sibling folder of SBB folders NeTEx, OJP and SIRI
2. A new folder `xcore-works` is created which is a sibling folder of folders NeTEx, OJP and SIRI.
3. The command-line calls are executed in folder `xcore-works`

Attention: the call examples contain linefeeds for readability. The linefeeds must be removed!

E1: contab report of OJP (variant 1)

Style: one domain for each target namespace

Call:

```
basex -b report=contab
      -b dir=../OJP
      -b odir=output/ojp
      -d "dnamesExcluded=.git .deprecated to%20be%20removed"
      -b custom=../xcore/custom-ojp.xml
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/ojp/contab/contab-index.html`

E2: contab report of OJP (variant 2)

Style: one domain for each XSD

Call:

```
basex -b report=contab
      -b dir=../OJP
      -b odir=output/ojp-per-xsd
      -b custom=../xcore/custom-ojp-perxsd.xml
      -b domains=xsd
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/ojp-per-xsd/contab/contab-index.html`

E3: contab report of SIRI

Style: one domain for each XSD

Call:

```
basex -b report=contab
      -b dir=../SIRI
      -b odir=output/siri-per-xsd
      -b custom=../xcore/custom-siri-perxsd.xml
      -b domains=xsd
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/siri-per-xsd/contab/contab-index.html`

E4: contab report of NeTEx

Style: one domain for each XSD

The report is created in two steps in order to avoid resource shortage.

```
basex -b report=edesc
      -b dir=../NeTEx
      -b odir=output/netex-per-xsd
      -b custom=../xcore/custom-netex-perxsd.xml
      -b domains=xsd
      ../xcore/xcore.xq
basex -b report=contab
      -b dir=../NeTEx
      -b odir=output/netex-per-xsd
      -b custom=../xcore/custom-netex-perxsd.xml
      -b domains=xsd
      -b edescReportDir=output/netex-per-xsd/edesc
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/netex-per-xsd/contab/contab-index.html`

Usage examples – `edesc` reports

The following examples assume:

1. `xcore` is installed as a sibling folder of SBB folders NeTEx, OJP and SIRI
2. A new folder `xcore-works` is created which is a sibling folder of folders NeTEx, OJP and SIRI.
3. The command-line calls are executed in folder `xcore-works`

Attention: the call examples contain linefeeds for readability. The linefeeds must be removed!

E1: edesc reports of OJP (variant 1)

Style: one domain for each target namespace

Call:

```
basex -b report=edesc
      -b dir=../OJP
      -b odir=output/ojp
      -b custom=../xcore/custom-ojp.xml
      ../xcore/xcore.xq
```

Report files: `xcore-works/output/ojp/edesc/*.xml`

E2: edesc reports of OJP (variant 2)

Style: one domain for each XSD

Call:

```
basex -b report=edesc
      -b dir=../OJP
      -b odir=output/ojp-per-xsd
      -b custom=../xcore/custom-ojp-perxsd.xml
      -b domains=xsd
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/ojp-per-xsd/edesc//*.xml`

E3: edesc report of SIRI

Style: one domain for each XSD

Call:

```
basex -b report=edesc
      -b dir=../SIRI
      -b odir=output/siri-per-xsd
      -b custom=../xcore/custom-siri-perxsd.xml
      -b domains=xsd
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/siri-per-xsd/edesc//*.xml`

E4: edesc report of NeTEx

Style: one domain for each XSD

Call:

```
basex -b report=edesc
      -b dir=../NeTEx
      -b odir=output/netex-per-xsd
      -b custom=../xcore/custom-netex-perxsd.xml
      -b domains=xsd
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/netex-per-xsd/edesc//*.xml`

Usage examples – `def` reports

The following examples assume:

1. `xcore` is installed as a sibling folder of SBB folders NeTEx, OJP and SIRI
2. A new folder `xcore-works` is created which is a sibling folder of folders NeTEx, OJP and SIRI.
3. The command-line calls are executed in folder `xcore-works`

Attention: the call examples contain linefeeds for readability. The linefeeds must be removed!

E1: Display the type definition "FareResultStructure" on the console

Style: Annotations retained

Call:

```
basex -b report=def
      -b dir=../OJP
      -b tnames=fareresultstr*
      -b skipAnno=0
      -s indent=yes
      ../xcore/xcore.xq
```

E2: Display the type definition without annotations

Style: Annotations removed

Call:

```
basex -b report=def
      -b dir=../OJP
      -b tnames=fareresultstr*
      -b skipAnno=1
      -s indent=yes
      ../xcore/xcore.xq
```

E3: Display the definitions of all attribute groups

Style: Annotations removed

Call:

```
basex -b report=def
      -b dir=../OJP
      -b hnames=*
      -b skipAnno=1
      -s indent=yes
      ../xcore/xcore.xq
```

E4: Write the definitions of all attribute groups into a file

Style: Annotations removed

Call:

```
basex -b report=def
      -b dir=../OJP
      -b odir=output/ojp
      -b ofile=attgroups.xml
      -b hnames=*
      -b skipAnno=1
      -s indent=yes
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/ojp/def/attgroups.xml`

*E5: Write a file containing the types *fare**

Style: (1) Annotations removed, (2) single report file

Call:

```
basex -b report=def
      -b dir=../OJP
      -b odir=output/ojp
      -b ofile=types-fare.xml
      -b tnames=*fare*
      -b skipAnno=1
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/ojp/def/types-fare.xml`

E6: Write a file containing all global attribute declarations

Style: (1) Annotations removed, (2) single report file

Call:

```
basex -b report=def
      -b dir=../OJP
      -b odir=output/ojp
      -b ofile=atts-all.xml
      -b anames=*
      -b skipAnno=1
      ../xcore/xcore.xq
```

Report file: `xcore-works/output/ojp/def/atts-all.xml`

*E7: Write a file containing the group definitions *passenger**

Style: (1) Annotations removed, (2) single report file

Call:

```
basex -b report=def
      -b dir=../OJP
      -b odir=output/ojp
      -b ofile=groups-passenger-with-anno.xml
      -b gnames=*passenger*
      -b skipAnno=1
      ../xcore/xcore.xq
```

Report files: `xcore-works/output/ojp/def/groups-passenger-with-anno.xml`

E8: Write all group definitions into report files, one file per XSD

Style: (1) Annotations removed, (2) one report for each XSD containing attribute groups

Call:

```
basex -b report=def
      -b dir=../OJP
      -b odir=output/ojp-groups
      -b domains=xsd
      -b gnames=*
      -b skipAnno=1
      ../xcore/xcore.xq
```

Report files: `xcore-works/output/ojp-groups/def//*.xml`

Tool output examples

Example tool output can be created by following the instructions in Usage examples – contab reports. Tool output can also be downloaded from here, along with source code and customization data (link expires 2023-09-01):

https://send.parsqube.de/file/C1NhdcU6AtdjVNfJ/8jtoCeXyf9S3qwgq/xcore-works.20230802.zip

Where to find what:

Path	Resource
output/ojp/contab/ojp.html	OJP report corresponding to previous OJP report
output/ojp/contab/contab-index.html	Entry point to OJP reports
output/siri/contab/contab-index.html	Entry point to SIRI reports
output/netex/contab/contab-index.html	Entry point to NETEX reports
xcore-custom/*	Customization files used to create the reports
xcore-sourcecode	xcore source code

The **OJP schemas** are reported using *one HTML file per target namespace.* (This holds only approximately – the OJP namespace report also contains a couple of SIRI components, in order to mimick the old generation report precisely.) In order to inspect the report, you may either open `ojp.html` (with contents corresponding to the old generation report), or `contab-index.html`, with a TOC giving access to all namespace reports. You may of course also immediately open another HTML file corresonding to a different target namespace (e.g. `ifopt.html`).

The **SIRI** and **Netex** reports consist of *one HTML file for each XSD.* An index page (`contab-index.html`) displays a TOC with links to all XSD reports.

contab report

The report is a slightly extended version of the report hitherto produced using the old generation tools `ojp/docs/ojp-*.xsl`.

Differences

The main differences are summarized by the following table.

New Behavior	Old Behavior
Generic and customizable	Not generic, not customizable
All type and group references can be navigated	Only references in the OJP namespace
Multiple HTML files – one for each target namespace	One single HTML output file
Complex type description includes base type contents	Base type contents is not shown
Also local type content is reported	Local type content is ignored
Complex type restriction are described correctly	Complex type restriction was not recognized
Description of enumeration types includes the documentation of the individual enumeration values	Enumeration types reported without documentation
Uses an intermediate format which is purely structural information without commitment to layout.	Uses an intermediate format designed to support the layout of the final report.

Some details

Generic and customizable

The xcore application is a **generic** XSD processing tool, without any assumptions concerning the style of the XSDs to be evaluated. Note however the following limitations:

(a) The following XSD 1.1 feature is not used: `alternative`
(b) The `xs:redefine` feature is not used
(c) No use of chameleon schemas (schema with a target namespace including a schema without target namespace)

Only some customization is supported by call parameters. The main part of customization is enabled by the use of a **customization file**, passed to the application via call parameter (`$custom`) and containing the details of customization. For example, the use of namespace prefixes can be controlled by customization. As another example, display names of schema components can be edited, a feature used in order to reproduce the name editing behaviour of the old generation report tool (omitting the name suffix “Group” and “Structure” in certain contexts).

Unlimited navigation

The old generation report did not contain links to components across target namespace boundaries. More precisely, only types and groups within the OJP namespace were accessible. The limitation was probably motivated by the wish to restrict the documentation to OJP contents, rather than include the extensive contents of other namespaces, especially the SIRI namespace.

The new generation report supports unlimited navigation without extending the documentation of the OJP namespace. This is achieved by producing a distinct report files for different target namespaces, connected by links.

Multiple HTML files

The possibility to distribute the schema report over several or even many HTML files is crucial for reporting large schema systems.

By default, only a single HTML report is created. The creation of multiple HTML files is controlled by a parameter (`$domains`) or by the customization document. The report of the OJP schemas was created using a customization reproducing the contents of the old report. Such behaviour is achieved using “custom domains”, with precise information which XSDs to include and in which order to display their reports.

Note that the very large system “NeTEx” was reported by using a different feature - “one HTML file per XSD”.

Appendix: example of an edesc representation

The following listing shows an example of the edesc ("expanded component descriptor") of a complex type definition. Note that type content is structured in a logical way without commitment to a tabular representation. Some points of possible interest:

- Base type references are (recursively expanded)
- Group references are (recursively expanded)
- Types of attributes or child elements are categorized (`@typeCategory`)
- Simple types of attributes or child elements are described in detail (`@typeDesc`)
- Component names and references use normalized prefixes, which are the same across all XSDs of the system
- All content items (attributes, elements, groups) have an `@z:occ` attribute providing a standardized representation of cardinality constraints

```
<z:complexType z:name="siri:ProducerResponseStructure" z:typeCategory="cc"
               xml:base="file:///C:/projects/sbb/ojp/siri/xsd/siri/siri_requests.xsd">
  <z:baseType z:name="siri:ResponseStructure" z:typeCategory="cc">
    <z:sequence z:occ="1:1">
      <z:element z:name="siri:ResponseTimestamp" z:reference="yes" z:elementFromCache="yes"
                 z:occ="1:1" z:type="xs:dateTime" z:typeCategory="sb" z:typeDesc="xs:dateTime"
                 xml:base="file:///C:/projects/sbb/ojp/siri/xsd/siri/siri_request_support.xsd"/>
    </z:sequence>
  </z:baseType>
  <z:extension z:name="siri:ProducerResponseStructure">
    <z:sequence z:occ="1:1">
      <z:group z:name="siri:ProducerResponseEndpointGroup" z:reference="yes" z:groupFromCache="yes"
               z:occ="1:1">
        <z:sequence z:occ="1:1">
          <z:element z:name="siri:ProducerRef" z:type="siri:ParticipantRefStructure"
                     z:typeCategory="cs" z:minOccurs="0" z:occ="0:1"/>
          <z:element z:name="siri:Address" z:type="siri:EndpointAddress" z:typeCategory="se"
                     z:typeDesc="xs:anyURI: (empty restriction)" z:minOccurs="0" z:occ="0:1"/>
          <z:element z:name="siri:ResponseMessageIdentifier"
                     z:type="siri:MessageQualifierStructure" z:typeCategory="cs" z:minOccurs="0"
                     z:occ="0:1"/>
          <z:element z:name="siri:RequestMessageRef" z:type="siri:MessageRefStructure"
                     z:typeCategory="cs" z:minOccurs="0" z:occ="0:1"/>
        </z:sequence>
      </z:group>
      <z:group z:name="siri:DelegatorEndpointGroup" z:reference="yes" z:groupFromCache="yes"
               z:occ="1:1">
        <z:sequence z:occ="1:1">
          <z:element z:name="siri:DelegatorAddress" z:type="siri:EndpointAddress"
                     z:typeCategory="se" z:typeDesc="xs:anyURI: (empty restriction)"
                     z:minOccurs="0" z:occ="0:1"/>
          <z:element z:name="siri:DelegatorRef" z:type="siri:ParticipantRefStructure"
                     z:typeCategory="cs" z:minOccurs="0" z:occ="0:1"/>
        </z:sequence>
      </z:group>
    </z:sequence>
  </z:extension>
</z:complexType>
```