OSM XML

Available languages Help

• Deutsch • **English** • français • italiano • •

Other languages — Help us translating this wiki show

Main Page
The map
Map Features
Contributors
Help
Blogs
Shop
Donations
Recent changes

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information
Cite this page

Basics

XML is a so called meta format to provide even human readable data interexchange formats. Various file formats use this HTML tree like structures to embedd their datas like SVG, ODT,...

pros	cons
• human readable because of clear structure • machine independent due to exact definitions, e.g. character sets, XML schema definitions, DTD,... • ready to use parsers for general XML that can be customized for a concrete file format • good compression ratio	• very huge files • might need to decompress before • parsing takes a lot of time

OSM XML file format

The major tools in the OSM universe use an XML format following this XML schema definition that was first used by the API only. Basically it is a list of instances of our data primitives (nodes, ways, and relations) that are the architecture of the OSM model.

Here is a shortened example of a complete OSM XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.2524800"/>
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636" tim
 <node id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter" uid="36744" visible="true" version="1" changeset="323878" tim
 <node id="1831881213" version="1" changeset="12370172" lat="54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible="true" time
  <tag k="name" v="Neu Broderstorf"/>
  <tag k="traffic_sign" v="city_limit"/>
 </node>
 ...
 <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636" tim
 <way id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset="4142606" timestamp="2010-03-16T11:47:08Z">
  <nd ref="292403538"/>
  <nd ref="298884289"/>
  ...
  <nd ref="261728686"/>
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower Straße"/>
 </way>
 <relation id="56688" user="kmvar" uid="56190" visible="true" version="28" changeset="6947637" timestamp="2011-01-12T14:23:49Z">
  <member type="node" ref="294942404" role=""/>
  ...
  <member type="node" ref="364933006" role=""/>
  <member type="way" ref="4579143" role=""/>
  ...
  <member type="node" ref="249673494" role=""/>
  <tag k="name" v="Küstenbus Linie 123"/>
  <tag k="network" v="VVW"/>
  <tag k="operator" v="Regionalverkehr Küste"/>
  <tag k="ref" v="123"/>
  <tag k="route" v="bus"/>
  <tag k="type" v="route"/>
 </relation>
 ...
</osm>
```

See Data Primitives for details of the object categories.
See Map features about how real world objects are modeled and categorized.

The structure is the following:

- an XML suffix introducing the UTF-8 character encoding for the file
- an osm element, containing the version of the API (and thus the features used) and the generator that distilled this file (e.g. an editor tool)
  - a block of nodes containing especially the location in the WGS84 reference system
    - the tags of each node

- a block of ways
  - the references to its nodes for each way
  - the tags of each way
- a block of relations
  - the references to its members for each relation
  - the tags of each relation

See the /XSD, and /DTD pages for details of attempts to define the format in those languages.

Assumptions

If you develop tools using this format, you can be sure that:

- blocks come in this order
- bounds will be on API and JOSM data

You can *not* be sure that:

- blocks are there (e.g. only nodes, no ways)
- blocks are sorted
- element IDs are non negative (Not in all osm files. Negative ids are used by editors for new elements)
- elements have to contain tags (Many elements do not. You will even come across Untagged unconnected nodes)
- visible only if false and not in Planet.osm
- id or user name present (Not always, due to anonymous edits in a very early stage)
- Changesets have an attribute num_changes for (This was abandoned from the history export tool because of inconsistencies)
- version ordering is sequential (doesn't have to be)

JOSM uses an 'action' attribute instead of timestamp, version or changeset for new objects

Some flavours might have other restrictions.

Tools

See Planet.osm and Import, Export, Convert

Flavours

There are a few different file formats currently in use, all with slightly different goals.

- API
- JOSM file format
- osmChange
- planetdiff
- Osmdiff

JOSM file format

Main article: JOSM file format

The file format was designed by the author of JOSM. It basically is a logical extension of the data sent from the server. What it adds is an indication of the origin of the data and the bounding box it comes from (if possible). It is actually more of a storage format of data downloaded along with changes made by the user.

pros	cons
• Supports placeholders • Indicates the source of the data	• not streamable, must read the whole file prior to applying

Supported by:

- JOSM
- Bulk_upload.py (read-only)
- [library]
- osmconvert
- osmfilter

osmChange

Main article: OsmChange OsmChange is a file format was created by the author of osmosis and is a more general format for representing changes.

pros	cons
• Streamable When sorted properly this file is a continuous stream of changes that can be played in order. In osmosis the option **--sort-change** will put the change into streamable order.	• Doesn't indicate source of data

Placeholders are proposed as an extension though they are not widely supported.

Supported by:

- Osmosis
- Bulk_upload.py (read-only)
- [library]
- osmconvert
- osmchange (read-only)
- osmfilter

Technical features of change formats

This a list of things that are desirable in a change file format for exampe change sets

placeholders

Placeholders are a feature where objects that are created in the file can be used in the creation of the objects that depend on them. So a single file can create two nodes and join them with a segment without knowing beforehand the final IDs.

indication of origin

IDs used in OSM files can not be shared between servers. IDs are allocated by the server, not by clients. Thus it is useful if the change file indicates the source of any IDs used in the file.

streamable

Whether the file can normally be processed in a stream, without breaking referential integrity.

See also

- Daily update an OSM XML file

Categories: DataFormats | Development | Outputs | OSM API