Taplint standards coverage

Mark Taylor (Bristol)

IVOA Interop
Cape Town

9 May 2016

$Id: taplint.tex,v 1.7 2016/05/05 10:14:18 mbt Exp $

taplint

TAP validation suite

- Point it at a TAP service, it tells you what's non-compliant or questionable
- Part of STILTS package
- Runs on the command line, reports output to stdout
- Mainly intended for TAP service developers/operators
- Output intended for humans, but somewhat machine-readable too

Availability:

- Documentation:

  http://www.starlink.ac.uk/stilts/sun256/taplint.html
  (but some output may require explanation from me, or examining the source code)

- Download:

  stable: http://www.starlink.ac.uk/stilts/
  latest: ftp://andromeda.star.bris.ac.uk/pub/star/stilts/pre/stilts.jar

Coverage Overview

TAP is complicated

- Embraces lots of different standards; at least:
  - **ADQL**: query specification
  - **VODataService**: metadata specification
  - **VOSI**: generic VO service description
  - **TAPRegExt**: TAP-specific service description
  - **VOTable**: table data serialization (result and upload)
  - **UWS**: asynchronous job control
  - **DALI**: generic data access service behaviour
  - **TAP**: metadata model, putting it all together
  - **ObsCore**: observation data model
  - **VOUnits**: metadata constraints
  - **UCD**: metadata constraints
  - **RegTAP**: relational registry data model

  list is not comprehensive; some standards rely on more basic ones e.g. VOResource
- Validating a TAP service relies on validating some/all of the above
- taplint tries (imperfectly) to do it

Example

```
% stilts taplint http://example.com/tap

Section TMV: Validate table metadata against XML schema
I-TMV-VURL-1 Validating http://example.com/tap/tables against http://www.ivoa.net/xml/VODataService/VODataServ...
E-TMV-URBH-1 (l.2, c.178): cvc-elt.1: Cannot find the declaration of element 'vosi:tableset'.
E-TMV-BBUE-1 (l.889, c.58): cvc-complex-type.2.2: Element 'dataType' must have no element [children], and the ...
E-TMV-BBUE-2 (l.893, c.58): cvc-complex-type.2.2: Element 'dataType' must have no element [children], and the ...
E-TMV-BBUE-3 (l.897, c.58): cvc-complex-type.2.2: Element 'dataType' must have no element [children], and the ...
E-TMV-BBUE-x (1817 more)
S-TMV-VALI-1 warnings: 0, errors: 1818, fatal: 0

Section TMS: Check content of tables metadata from TAP_SCHEMA
I-TMS-QSUB-1 Submitting query: SELECT principal, indexed, std, "size" FROM TAP_SCHEMA.columns
I-TMS-QGET-1 Query GET URL: http://example.com/tap/sync?REQUEST=doQuery&LANG=ADQL&QUERY=SELECT+principal%2C+in...
E-TMS-RRES-1 TAP response document RESOURCE element is not marked type='results'
E-TMS-CERR-1 Error reading TAP_SCHEMA.columns data [IllegalArgumentException: Column ["size"] does not exist.]
   . . .
```

Notes on output:

- Output intended to be read by humans
- Manageable length (repeated messages filtered out)
- *Tries* to give as much info as possible
- Logs queries as well as reporting errors where possible/practical to aid reproducibility
- Options available for truncating long lines, filtering messages, restricting tests, etc
- Programmatic use is also possible (output is `grep`-friendly, semi-public API exists)

Stages

Tests performed:

- `TMV`: Validate `/tables` endpoint against VODataService schema
- `TME`: Check content `/tables` endpoint for consistency
- `TMS`: Check form and content of TAP_SCHEMA tables
- `TMC`: Compare table metadata from `/tables` and TAP_SCHEMA
- `CPV`: Validate `/capabilities` endpoint against TAPRegExt schema
- `CAP`: Check content of `/capabilities` endpoint
- `AVV`: Validate `/availability` endpoint against VOSI schema
- `QGE`: Make example ADQL queries in sync GET mode
- `QPO`: Make example ADQL queries in sync POST mode
- `QAS`: Make example ADQL queries in async mode
- `UWS`: Test asynchronous UWS/TAP behaviour
- `MDQ`: Check table query result columns against declared metadata
- `UPL`: Make example queries with table uploads
- `OBS`: ObsCoreDM data model

Validation activities by **Standard**:

Service Availability

VOSI

- Read availability document at `/availability` endpoint
- Validate XML against VOSIAvailability schema

Service Capabilities

VOSI

- Read capabilities document at `/capabilities` endpoint
- Validate XML against VOSICapabilities schema

TAPRegExt

- Validate XML against TAPRegExt schema
- Check declared query languages
  - ADQL 2.0 must be present
  - Check language features syntax
  - Check standard language features match standard
- Check upload declarations
- Check output format declarations
- Check data model declarations, trigger DM-specific validation accordingly

Job Submission

UWS

- Test job lifecycles:
  - ▷ create/delete; create/abort/delete; create/run, wait for completion
- Check job statuses at different lifecycle stages
- Check job list
- Validate job document
- Check various UWS endpoints are consistent with job document
- Test parameter (re)set/read.
- Check syntax of UWS fields (quote, executionDuration etc)
- Check MIME type of resources

Data Models

ObsCore

- Test whether ObsCore data model is declared correctly
  (including test for pre-erratum TAPRegExt wrong ObsCore ivoid)
- Check table name ivoa.ObsCore
- Check mandatory columns present
- Check column metadata: datatype, UCD, Utype, units
- Check column content: range, legal options, known options, illegal null values
- Supports ObsCore v1.0 (public release) or v1.0/v1.1 (currently pre-release)

TAP

- Test TAP_SCHEMA required tables and columns exist
- Check column datatypes
- Check foreign key links are not broken

Executing Queries

ADQL

- Queries against TAP_SCHEMA tables, declared tables, non-existent tables
- Makes basic single-table queries: `SELECT *`, `SELECT <col-list>`, `TOP`, `AS`, ...

  — *very incomplete coverage of ADQL features!*

TAP

- Runs all tests using sync GET, sync POST and async
- Checks `SELECT *` returns column list/types as declared in TAP metadata
- Checks correctly packaged error responses from duff queries
- Checks row overflow signalled correctly
- Checks `MAXREC=0` returns metadata-only table
- Checks output has correct MIME type
- Upload queries:
  - ▷ Check metadata/data content preservation on round trips
  - ▷ Uses TABLEDATA and BINARY upload serializations
  - ▷ Uses all different column types
  - ▷ Doesn't try to break it, too easy to do (weird column names?)

Table Output

VOTable

- Taplint (re-)uses STILTS `votlint` code for VOTable validation
- Detect declared/assumed VOTable version
- Check TAP output is required version (VOTable 1.2+)
- Version-specific validation of all known versions (1.0, 1.1, 1.2, 1.3)
- Validate against relevant schema (or DTD)
- Handle all known serializations (TABLEDATA, BINARY, BINARY2, FITS)
- Report deprecated elements/attributes
- Report on incorrect namespaces (but validate anyway)
- Check `ID`/`IDREF`s (links resolve, no repeated IDs, links meaningful)
- Warn about duplicated column names
- Check `TD` element contents, `PARAM/@value` attrs match datatype/arraysize declarations
- Check syntax for real/integer/boolean/array values
- Check nulls are represented legally
- Check `TD` count is as expected in each row
- Check reported `nrows` count is correct

Table Metadata

VOSI

- Read tables document at `/tables` endpoint (currently VOSI 1.0 only)
- Validate XML against VOSITables schema

VODataService

- Additional checks of XML against VODataService schema;
  warn but tolerate unknown XML namespaces
- Check for duplicate schema, table or column names
- Check legal syntax for table and column names
- Check contents match TAP_SCHEMA metadata declaration
  (schemas, tables, columns, foreign keys)

Notes

taplint operation:

- Works for TAP 1.0. Will take some effort to fix for TAP 1.1 service endpoint reorganisation.
- Tries to continue with tests even if earlier ones fail (where possible)
- Schema validation isn't as easy as it sounds
  - Use local copies of IVOA schemas
  - Continue validation in presence of missing/incorrect namespaces
  - Allow imported namespaces
- No other TAP validators - single point of failure?

Summary

Coverage of TAP-relevant standards by taplint (STILTS v3.0-6):

Standard	Status	Comments
TAP	good	ADQL only; should do /examples endpoint
VOTable	good	uses `votlint`
TAPRegExt	good	
ObsCore	good	
UWS	good	1.0 only; some tests hard to automate
VODataService	medium	CatalogService, not Coverage
VOSI	medium	mostly as implied by TAP
DALI	medium	mostly as implied by TAP
ADQL	basic	scope for many more tests
VOUnits	none	include in future versions?
UCD	none	include in future versions?
RegTAP	none	include in future versions?

Use of validation code outside STILTS/taplint:

- Some taplint stages can be run on their own
- There are some undocumented entry points (main methods)
- `votlint` is a standalone item for VOTable validation
- It's possible to write a programmatic (java) harness to incorporate `taplint` or `votlint` in other validators