Taplint:
TAP Service Validator

University of BRISTOL

Mark Taylor m.b.taylor@bristol.ac.uk

Introduction

TAP, the Table Access Protocol, is a Virtual Observatory protocol suite allowing client software to interact with remote database services in a standardised way, including acquiring rich metadata and submitting simple or complex queries in an SQL-like language. It underlies much current access to astronomy data archives, providing very powerful query capabilities, and is a VO success story. The protocol stack involved however is quite complex, involving a dozen or so separate IVOA specifications, so implementors and deployers face many opportunities to make mistakes.

Taplint, part of the STILTS package, is a suite of tests intended to assist TAP service operators to check whether a TAP instance is behaving in compliance with the various applicable standards. We present here an overview of its capabilities and usage.

Why Validate?

The Document Standards defined by the International Virtual Observatory Alliance (IVOA) require that a validator tool, as well as interoperable implementations, must be available before a standard can be endorsed. This is partly as a check of the written specification (implementing a validator is a good way to pick up ambiguities and inconsistencies in the text) and partly as an aid to both implementors and users of the protocols. Running validation tests on a service, ideally as part of the development and deployment process, can identify errors and issues that could otherwise cause trouble for service users.

Usage and Availability

Taplint comes as part of the STILTS table manipulation package, which is written in Java. To run it, you just need a Java runtime environment (Java 8 or later) and the stilts.jar file. Point it at the URL of the TAP service to test, and it will write reports to standard out:

```
    java -jar stilts.jar taplint tapurl=http://example.com/tap
```

You can optionally specify various parameters to control which testing stages are run, the form of the output, details of service interaction, etc.

Output is by default line-oriented, grep-friendly, and intended for human consumption. Each output line includes a level code, a 4-character ID indicating the test being run (there are currently 200+ of these), and an informative message. The main Levels are:

- [I]nfo — *information about the test being performed*
- [W]arning — *behaviour that is questionable or not recommended by standards*
- [E]rror — *behaviour contrary to standards requirements*

Steps are taken to limit the output length, so you won't see hundreds of reports of the same problem even if multiple similar failures are encountered.

Considerable efforts are made to explain the nature of the issues reported. However, failures can be subtle and bugs are possible, so users are encouraged to contact the author with any questions about taplint output, behaviour, or anything else.

Full details are available in the STILTS documentation.

Standards Covered

The TAP standard itself defines how clients can interrogate metadata, pose queries, and retrieve results from remote database services. However it does this with reference to many other IVOA (and some non-IVOA) specifications concerning serialization, service interaction, metadata encoding, capability declaration and more; there are also further standards defining domain-specific data models that sit on top of TAP. Taplint understands most of these, and runs specific tests that a target service is behaving as prescribed.

Behaviour defined by the following IVOA specifications is tested to a greater or lesser extent: TAP, VOTable, UWS, VODataService, ADQL, VOResource, VOSI, TAPRegExt, DALI, ObsCore, ObsLocTAP, EPN-TAP, UCD, VOUnits, SoftID and SSO,

These documents, available from `https://www.ivoa.net/documents/`, are the result of much collaborative work over many years between IVOA members.

Tests Performed

Taplint runs a battery of test queries against the TAP service, posing as a client and checking that the response in each case is as mandated by the relevant specifications. It tries to test as many aspects of the required behaviour as it can, especially in the areas of metadata provision, output formats, job submission and compliance with data models.

The tests are grouped in a sequence of stages; by default it runs all that are applicable to the target service, but you can tell it to run only a few if you're interested in particular behaviour. The stages currently available are:

- `TMV`: Validate table metadata against XML schema
- `TME`: Check content of tables metadata from /tables
- `TMS`: Check content of tables metadata from TAP_SCHEMA
- `TMC`: Compare table metadata from /tables and TAP_SCHEMA
- `UUC`: Check column units and UCDs are legal
- `CPV`: Validate capabilities against XML schema
- `CAP`: Check TAP and TAPRegExt content of capabilities document
- `AVV`: Validate availability against XML schema
- `QGE`: Make ADQL queries in sync GET mode
- `QPO`: Make ADQL queries in sync POST mode
- `QAS`: Make ADQL queries in async mode
- `UWS`: Test asynchronous UWS/TAP behaviour
- `MDQ`: Check table query result columns against declared metadata
- `OBS`: Test implementation of ObsCore Data Model
- `LOC`: Test implementation of ObsLocTAP Data Model
- `EPN`: Test implementation of EPN-TAP tables
- `UPL`: Make queries with table uploads
- `EXA`: Check content of examples document

Note that it is not possible to test all conceivable inputs; just because taplint doesn't identify errors in a service does not guarantee that none are present.

Discussion

Taplint provides a way to identify flaws in running TAP services during development and operations, and can thus contribute to the efforts of service providers in improving the experience for their users. It has been used at many data centers for this purpose, including by ASDC, CADC, CDS, CfA, ESA, ESO, GAVO, IPAC, MAST and NASA, as well as providing the TAP-specific component of the bulk validation services run by ESA, PADC and NASA that assess overall VO operational status. If you are developing or operating a TAP service, please consider running taplint to check its compliance with the standards. If you want help doing that, please come and find me at ADASS or contact me afterwards.

The VO standards landscape is in constant flux, and taplint requires frequent updates in response to new and updated specifications, changing or contested interpretations of existing documents and user feedback. Recent improvements (STILTS v3.4-2) include a new UCD and Unit validation stage, validation of the data and metadata specified by the new EPN-TAP and ObsLocTAP data models, better stage selection control, checking service VO component identification, improved reporting of VOTable issues, and validation of content declared by xtype.

Writing validation tools like taplint is (like much of the work in the VO) time-consuming and not very glamorous, but they can provide an important contribution to the robustness of the service ecosystem, as well as feeding back to improve the standards definition process.

Acknowledgements: This work has been supported by STFC and GAVO.

Example Run

```
This is STILTS taplint, 3.4-2
Timestamp: 2021-10-12 17:34:06 BST
Static report types: ERROR(169), WARNING(76), INFO(30), SUMMARY(12), FAILURE(25)
...
Section TME: Check content of tables metadata from /tables
I-TME-CURL-1 Reading capability metadata from http://vlkb.neanias.eu:8080/vlkb/tap/capabilities
I-TME-TURL-1 Reading table metadata from http://vlkb.neanias.eu:8080/vlkb/tap/tables
E-TME-CRSV-1 Column name is ADQL reserved word 'distance' in table compactsources.sed_view_final - should delimit like '"distance"'
E-TME-CRSV-2 Column name is ADQL reserved word 'area' in table filaments.branches - should delimit like '"area"'
E-TME-CRSV-3 Column name is ADQL reserved word 'area' in table filaments.filaments - should delimit like '"area"'
E-TME-CRSV-4 Column name is ADQL reserved word 'size' in table TAP_SCHEMA.columns - should delimit like '"size"'
S-TME-SUMM-1 Schemas: 5, Tables: 25, Columns: 1057, Foreign Keys: 0
S-TME-FLGS-1 Standard column flags: indexed: 10, primary: 29, nullable: 0
S-TME-FLGO-1 Other column flags: none
...
Section UUC: Check column units and UCDs are legal
E-UUC-UCDX-1 Bad UCD "meta.ref.uri;meta.curation"; BAD_SEQUENCE: Too many (2) PRIMARY UCD words: "meta.ref.uri", "meta.curation"! Only one is allowed in a UCD.; 1 PRIMARY UCD word not in first position: "...
E-UUC-UCDX-2 Bad UCD "meta.id; obs.proposal"; BAD_SYNTAX: Wrong syntax for 1 UCD word: " obs.proposal" in column ivoa.obscore.proposal_id
...
```

A typical full run might take 3 minutes and generate 200 lines of output, though this can vary *a lot* depending on the service.