

# **STIL for (vO)Table Processing in Java**

Mark Taylor (University of Bristol)

European Data Providers Forum  
Online

November 2021

\$Id: stil.tex,v 1.5 2021/11/23 12:45:51 mbt Exp \$

# STIL Overview

## Starlink Tables Infrastructure Library

- Open source Java table I/O library
- I/O layer underlying STILTS and TOPCAT
- Format-neutral (mostly)
- ... but has some VOTable-specific capabilities
- *Fairly* stable API
- Various formats supported for input and/or output:
  - ▷ VOTable, FITS, ECSV, MRT, CDF, ASCII, CSV, IPAC, GBIN, Parquet, ...
- Full documentation <http://www.starlink.ac.uk/stil/>:
  - ▷ Tutorial introduction: [SUN/252](#)
  - ▷ Comprehensive [javadocs](#)
- Good user support (I claim): [topcat-user@jiscmail.ac.uk](mailto:topcat-user@jiscmail.ac.uk) or [m.b.taylor@bristol.ac.uk](mailto:m.b.taylor@bristol.ac.uk)

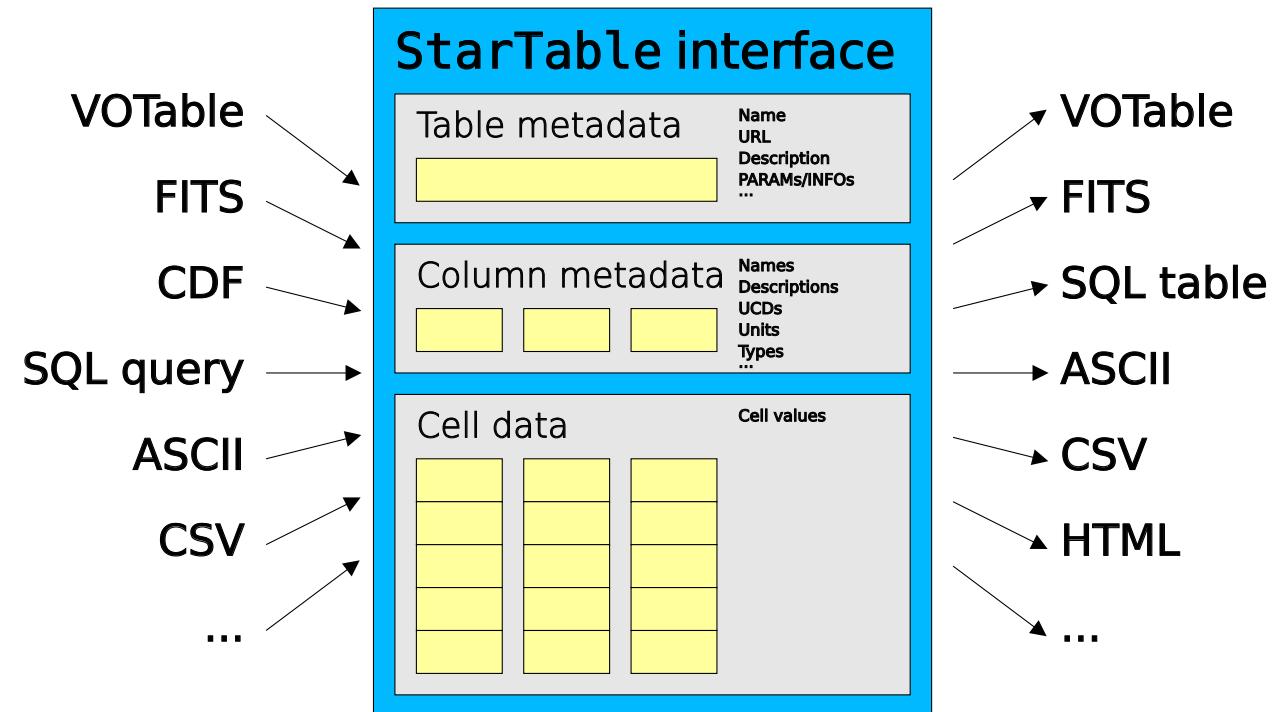
## Dependencies

- Java 8 (earlier STIL versions depended on earlier Java versions)
- `nom.tam.fits` (v0.9, v1.15.2, future versions?) — this can cause problems
- Not much else required except for niche I/O formats (ECSV, Parquet, Feather, ...)
- ~2 Mb total

# StarTable interface

- STIL represents a table as StarTable instance

- Per-table metadata
  - Name, Data type, Array size, Description, Units, UCD, custom items
- Dimensions
  - ▷ Column count
  - ▷ Row count (if known)
- Data cells in rows and columns
- Various ways to access data:
  - ▷ `getRowSequence`: Iterator over rows
  - ▷ `getCell/getRow`: Random access (if available)
  - ▷ `getRowSplittable`: Multi-thread-friendly iterator over rows (if available) — STIL 4.0
  - ▷ `getRowAccess`: Multi-thread-friendly random access (if available) — STIL 4.0



# StarTable I/O

- Input handlers turn an input source (e.g. file) into a StarTable
- Output handlers write a StarTable to a byte stream
- Manipulate StarTable objects without worrying about input file format
- I/O utility classes keep track of available handlers

```
StarTable t = new StarTableFactory().makeStarTable(fileName);
new StarTableOutput().writeStarTable(t, "file.xml", "votable");
... etc (streaming input also available)
```

- Format conversion is trivial
  - Read one format to StarTable
  - Write StarTable to another format
  - ... but metadata etc may not be preserved perfectly

# VOTable Handling

- You can use generic StarTable interface
  - VOTable-format input and output handlers are supplied
  - StarTable interface does not preserve all VOTable-specific information
    - ▷ GROUP, FIELDref, PARAMref, VO-DML?
- Some VOTable-specific read options also available:
  - Preserve VOTable document structure, but benefit from STIL data access
  - Full VOTable document structure visible using standard XML interfaces:
    - ▷ DOM: XML document tree in memory
    - ▷ SAX: XML document elements streamed
  - But memory usage is low
    - ▷ Metadata is treated using standard XML APIs
    - ▷ Bulk table data is read/streamed using efficient STIL-specific API
  - All works equally for all VOTable serializations (TABLEDATA, BINARY, BINARY2, FITS)
- Writing custom VOTable metadata
  - Use print statements to write custom elements (GROUPs, VO-DML, etc)
  - Use STIL to write core metadata and bulk data (VOTable FIELD/TABLE elements)

# Memory Management

Some tables are large

- Avoid having to store them in memory if at all possible
- Where possible use *file mapping* for input (disk read on-demand by OS)
  - ▷ Suitable for uncompressed binary data, e.g. FITS BINTABLE, CDF
- STIL streams data wherever possible
- Uses configurable StoragePolicy when streaming to random access
  - ▷ Can use memory, temporary disk file, or (default) adaptive combination

*It is usually possible to do what you want to in STIL with a small memory footprint*

# Why use STIL in the VO?

- Suitable if you're writing Java software manipulating (VO)Tables
- Good VOTable support
  - All VOTable versions supported (1.0, 1.1, 1.2, 1.3, 1.4)
  - All VOTable serializations supported (TABLEDATA, BINARY, BINARY2, FITS)
  - Tolerant of malformed VOTable input  
(extra/missing elements, broken namespaces etc ignored, as long as the basics are present)
  - Likely to track future changes in VOTable standard
- Support for lots of formats
  - Very easy to offer service users upload/download in VOTable, FITS, CSV, ECSV, ...
- Efficient
  - Streaming/low memory footprint even when handling large tables
- STIL is used by various existing VO service providers
  - VOLLT
  - ESDC TAP+ (GACS/Gaia)
  - ESO TAP
- Good user support

## Things I didn't mention

- Pluggability
  - Provide your own I/O handlers, StarTable implementations, ...
- Lazy data access
  - Table cells only read/calculated if actually used
- RDBMS access
  - Read/write to databases using JDBC
- Table *schemes*
  - E.g. generate arbitrary length tables of test data, simulated catalogues, ...  
(e.g. test data, simulated sky data)
- “FITS-plus”
  - FITS variant combining FITS efficiency with VOTable metadata
- ...

## Hands-On Demo

No.

- But see examples in <http://www.starlink.ac.uk/stil/sun252/>

## Further Information

- STIL web page: <http://www.starlink.ac.uk/stil/>
- Tutorial documentation: [SUN/252](#)
- API reference: [javadocs](#)
- Mailing list: [topcat-user@jiscmail.ac.uk](mailto:topcat-user@jiscmail.ac.uk)
- Talk to me: [m.b.taylor@bristol.ac.uk](mailto:m.b.taylor@bristol.ac.uk)