

# From the Bench to Orbit

Test-as-you-Fly processing with Das3

Kevin Steele

Chris Piker, Larry Granroth, David Miles

Department of Physics and Astronomy, University of Iowa

# Data Processing by Mission Phase

## NASA Phase B (Preliminary design/tech completion)

- Data displayed via O-scope, LabView, etc.
- Data artifacts usually ad-hoc



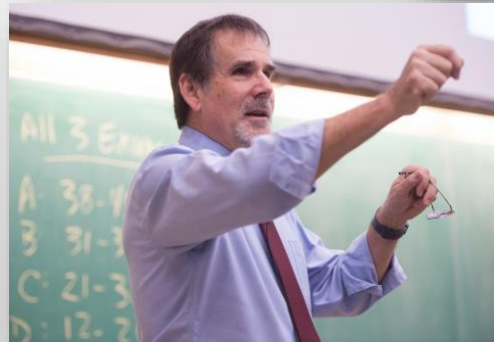
## NASA Phase C (Final design and Fabrication)

- Embedded CPUs are working
- Packetized data available
- Engineers need NeRT (Near Real Time) display
- Packet formats often in flux



## NASA Phase D (Assembly, Integration and Test)

- Scientists using flight-like science products
- NeRT display still essential



## NASA Phase E (Operations)

- Archive quality science product files
- NeRT display rarely applicable

# Data Processing by Mission Phase

## NASA Phase B (Preliminary design/tech completion)

- Data displayed via O-scope, LabView, etc.
- Data artifacts usually ad-hoc



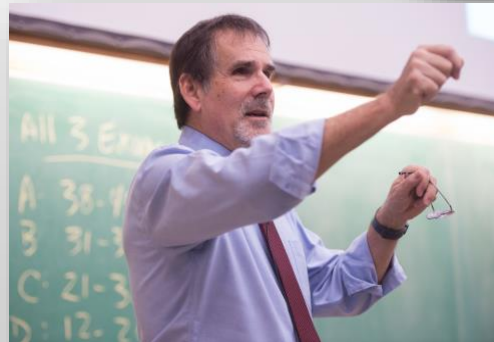
## NASA Phase C (Final design and Fabrication)

- Embedded CPUs are working
- Packetized data available
- Engineers need NeRT (Near Real Time) display
- Packet formats often in flux



## NASA Phase D (Assembly, Integration and Test)

- Scientists using flight-like science products
- NeRT display still essential

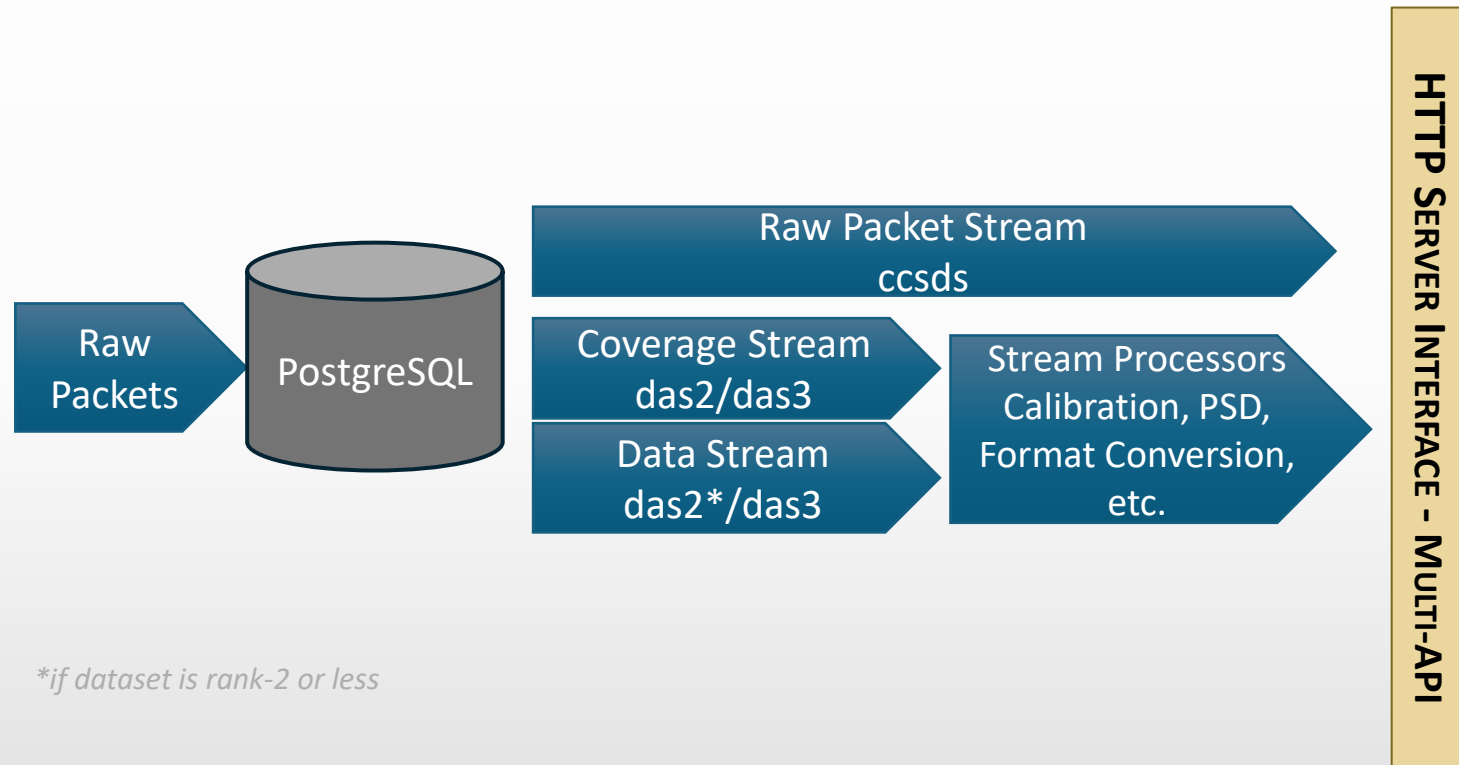


## NASA Phase E (Operations)

- Archive quality science product files
- NeRT display rarely applicable

Supported phases

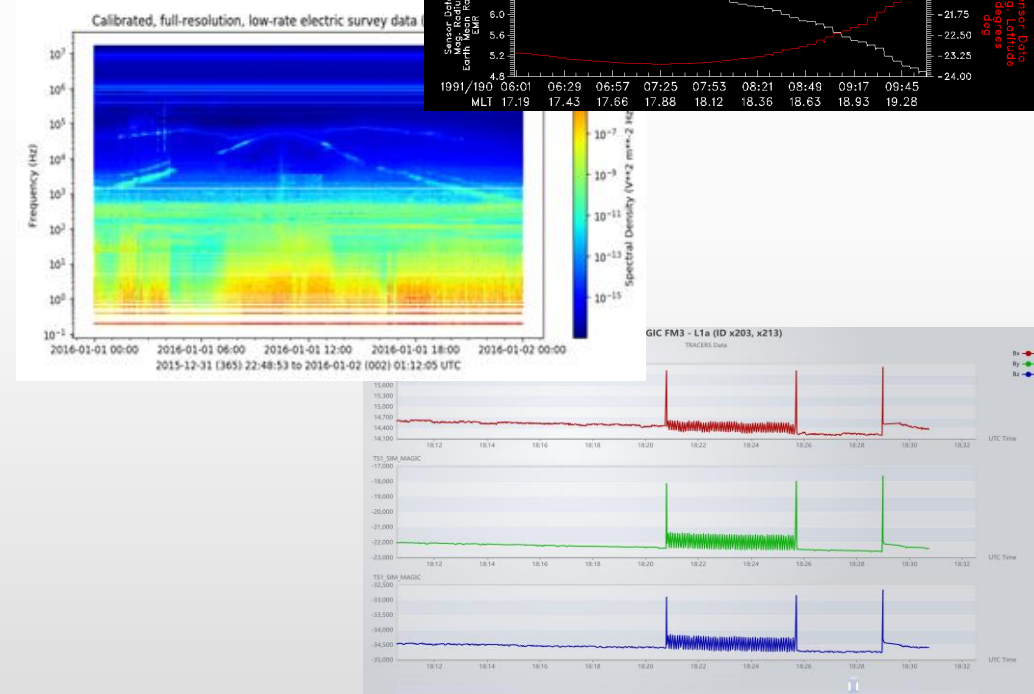
# Architecture as Implemented



*\*if dataset is rank-2 or less*

SERVERS

CLIENTS



# New Protocol: dasStream v3.0

- Used for inter-process communication.
- Cleanly separates physical dimensions from array dimensions
- Handles high-rank data, up to 7 independent coordinates.
- Does not assume rectangular arrays
- Handles text data and variable length items
- Header validation via [XSD schemas](#).

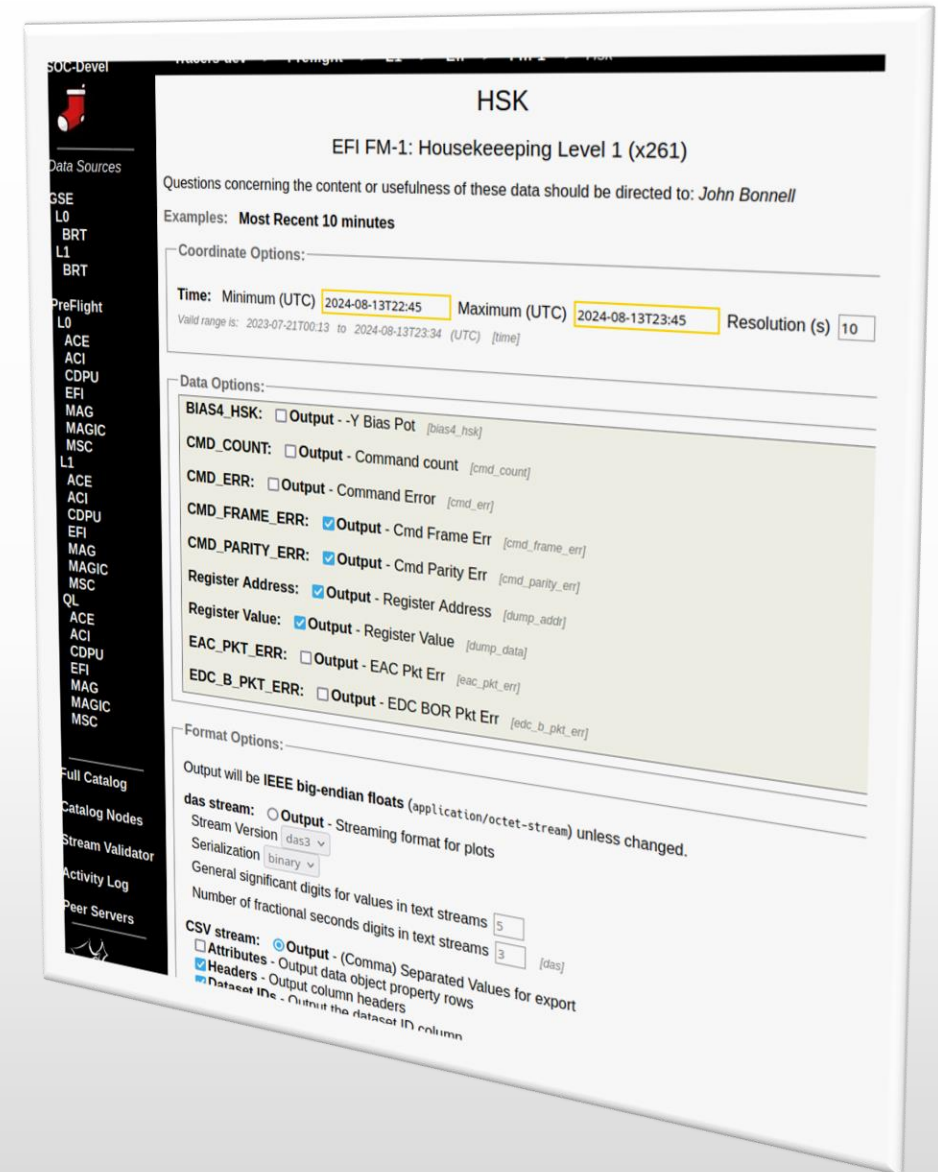
```
1 |Sx| |50|
2 <stream type="das-basic-stream" version="3.0">
3   <properties>
4     <p name="instrument_host">MEX</p>
5     <p name="instrument">MARSIS</p>
6   </properties>
7 </stream>
8 |Hx| |1|3455|
9 <dataset name="ais" rank="3" index="*;160;80" plot="cartesian3d" >
10
11 <coord physDim="time" axis="x">
12   <properties><p name="label">SCET</p></properties>
13   <scalar use="center" semantic="datetime" index="*;-;" units="UTC">
14     <packet numItems="1" itemBytes="24" encoding="utf8" />
15   </scalar>
16 </coord>
17
18 <coord physDim="altitude" axis="y">
19   <properties><p name="label">cΔt/2, Apparent altitude</p></properties>
20   <scalar use="reference" semantic="real" storage="float" index="*;-;" units="km">
21     <packet numItems="1" itemBytes="12" encoding="utf8" />
22   </scalar>
23   <scalar use="offset" semantic="real" storage="float" index="-;-;80" units="km">
24     <sequence minval="0.0000e+00" interval="-1.3705e+01" />
25   </scalar>
26 </coord>
27
28 <coord physDim="frequency" axis="z">
29   <properties><p name="label">Frequency (MHz)</p></properties>
30   <scalar use="center" semantic="real" storage="float" index="-;-;160;-;" units="MHz">
31     <values>
32       1.09400e-01 1.20500e-01 1.31200e-01 1.42300e-01 1.53000e-01 1.75200e-01 1.85900e-01
33       2.07600e-01 2.18800e-01 2.29900e-01 2.40500e-01 2.51700e-01 2.73400e-01 2.84600e-01
34       3.06300e-01 3.17000e-01 3.28100e-01 3.39200e-01 3.49900e-01 3.61000e-01 3.71700e-01
35       ...
36     </values>
37   </scalar>
38 </coord>
39
40 <data physDim="E_spec_dens" name="sounder_return">
41   <properties>
42     <p name="label">Spectral Density (V!a2!nm!a-2!nHz!a-1!n)</p>
43   </properties>
44   <scalar use="center" semantic="real" index="*;160;80" units="V**2 m**-2 Hz**-1" >
45     <packet numItems="12800" itemBytes="12" encoding="utf8" fill="-1.0e+31"/>
46   </scalar>
47 </data>
48 </dataset>
49 |Pd| |1|153636|2012-12-21T15:06:40.596 1.5404e+03 1.7882e-14 1.1964e-14 4.3845e-15 1.9897e-15
50 5.8294e-16 5.5629e-15 3.6921e-16 5.1378e-16 5.3454e-15 2.7233e-15 1.1994e-15 8.0604e-16
51 4.7716e-15 3.9284e-15 2.8748e-15 2.0690e-14 2.5603e-16 2.7896e-14 4.0383e-14 4.3642e-14
```

*A das3 stream, binary and text encodings are supported*



# New Server: dasFlex

- Is a backwards compatible to [das2py-server](#)
- Server core is format agnostic, each installed component indicates supported formats
- Reads HTTP GET parameters and solves for a command pipeline, then runs it for data.
- It's supplied with das2 and das3 components, but others can be installed.
- Can convert das2 or das3 pipeline output **CSV** or **CDF**.



# New Reader: dasTelem

- Provides bulk telemetry storage and data readers for dasFlex
- Removes the need to write custom data reader programs for early mission data.
- Written in D and PL/pgSQL. Relies on PostgreSQL for packet storage, parsing configuration and calibration data.
- Real-time events handled via PostgreSQL notifications
- Not yet optimized. Initial performance is 2.3 to 16.2 MB/sec, depending on packet complexity.



A portion of the dasTelem DB schema



# New Client: DASOC

- Web application parsing and utilizing das3 stream data for creating interactive displays
- NodeJS project using JavaScript/TypeScript primarily
- Graphing powered by Apache ECharts
  - Open-source, well documented, data sampling built in, WebGL extension for 3D graphing, and highly customizable
- Ongoing incorporation of Web-Assembly for performance
  - For smaller tasks JS is more than capable, but WASM scales better with data size





# DASOC Internals

- ViteJS used for building
  - Like create-react-app but without the webpack headaches
- React as frontend framework
- Redux helps with data state management
- Gitlab-CI Pipeline to run tests, check build, and generate Docker image automatically
- Currently running nginx Docker container for hosting
- Vitest to run unit testing



ViteJS



Vitest



Docker



Redux



Gitlab

# Suite Demo

The screenshot shows the 'Multistream Server' web interface. On the left is a vertical navigation menu with categories like 'Data Sources', 'PreFlight', 'Full Catalog', 'Catalog Nodes', 'Stream Validator', 'Activity Log', and 'Peer Servers'. The main content area is titled 'Multistream Server' and contains text explaining the server's purpose: 'This server provides data streams in a variety of formats using a variety of Application Programming Interfaces (APIs)'. It lists supported data formats (CSV, CCSDS, das3) and clients (Autoplot, SPEDAS, SDDAS). Below the text is a code block showing a directory tree structure for data sources.

DasFlex

The screenshot shows the 'DASOC Display Application for Science Operations Center v1.6.0'. The interface features a yellow header bar with the 'IOWA' logo and navigation links for 'Bookmarks' and 'Instrument Specific Displays'. A central panel displays a scientific visualization of magnetic field lines. To the left of this panel is an 'Options Menu' with expandable sections: 'Source Select' (containing a 'Sources Catalog' dropdown and a 'Realtime Streaming' checkbox), 'Direct URL Override' (with an input field), 'Query Parameters', 'Plot Options', and 'Socket Settings'. Each section has an 'Apply' or 'Reset Data' button. Above the visualization are buttons for 'Options', 'Share Plot Link', 'Edit Chart', and navigation arrows. Below the visualization is a welcome message: 'Welcome to the Display Application for Science Operations Center (DASOC)'.

DASOC

# Summary and Q&A

Open projects by late Spring 2025 to coincide with TRACERS launch

- Das Suite: <https://github.com/das-developers>
- Kevin Steele: [kevin-steele@uiowa.edu](mailto:kevin-steele@uiowa.edu)
- Chris Piker: [chris-piker@uiowa.edu](mailto:chris-piker@uiowa.edu)
- Larry Granroth: [larry-granroth@uiowa.edu](mailto:larry-granroth@uiowa.edu)
- Carrie Gonzalez @ SwRI
- Special thanks to David Miles @ U of I
- Special appreciation to Prof. Craig Kletzing