IN11D-0064: Lightweight Federated Data Networks with Das2 Tools

C.W. Piker[1], L.J. Granroth[1], J. Mukherjee[2], D. Pišа[3], B. Cecconi[4], A.J. Kopf[1], J.B. Faden[1] *1. University of Iowa, 2. Southwest Research Institute, 3. IAP CAS, Prague, 4. Observatoire de Paris, PSL, CNRS*

The University of Iowa — das2 http://das2.org

Das2 describes a collection of cooperating programs originally created to support daily review and analysis activities of the Cassini RPWS investigation. The system proved useful and is now relied upon for rapid access to working data sets from many planetary missions including Mars Express and Juno, heliophysics missions including Van Allen Probes and Voyager, as well as ground based radio astronomy results from the Nançay Decameter Array and Long Wave Array. As more teams have begun using das2 tools the need for updates to what was largely an in-house protocol and tool-set have become apparent.

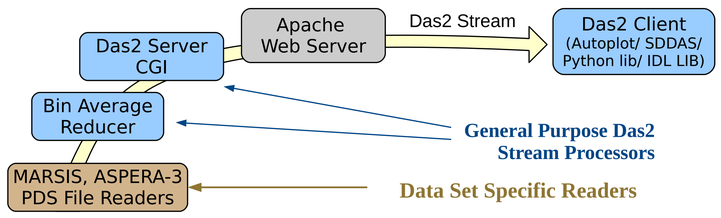
University of Iowa Space Physics Data Streaming Heritage

- ***das1 - 1994 to 2002:*** *Initial standardized data streams by Larry Granroth in support of Galileo PWS operations, browser based interface*
- ***das2 - 2002 to 2006***: Java clients written to support modern data navigation using computer mouse gestures, new stream format, server-side data reduction via low-overhead C programs
- ***Autoplot - 2005 to present***: Externally funded project, (autoplot.org) built on das2 Java libraries, added CDF reading, scripting and many other useful features.
- ***pyServer - 2012 to 2014***: Re-write of server from perl to python, das2 stream format documented, multi-resolution caching added.
- ***libdas2* - 2016 to present**: Upgrades to C tools, data model added, units handling, HAPI adapter, direct interface to NumPy and pure python upper layer.
- ***das2 catalogs - 2018***: Data source listings re-written in JSON and moved off-server to form a robust distributed network suitable for use as a public data source.

U. Iowa das2 servers currently host **>155 TB** of space physics related data from Voyager, Cluster, Cassini, Mars Express, Juno, Van Allen Probes, LWA-1 ground station and others.

Das2 Servers: Built to Support Rapid Display

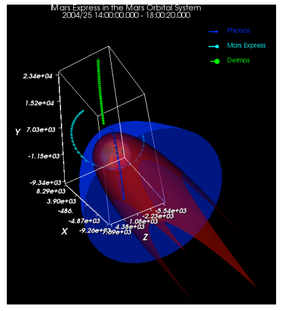
Since 2002 most space physics data at the University of Iowa have been provided to analysis software in the form of das2 streams. These are generated by data set specific programs called readers which translate the instrument specific files into a common stream output. When viewing data over time scales that are large compared to the intrinsic instrument resolution, streams can be reduced on the server before transmission to the display program, thus conserving network bandwidth and client memory. For particularly large or inefficient file sets, streams are cached on the server so that subsequent requests are handled in a timely manner.

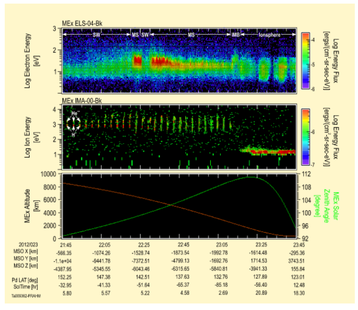


Most client applications in use by the Iowa Radio and Plasma Wave group are written Java. However many of our collaborators commonly produce software in other languages. To allow for easier data and software exchange, we have undertaken efforts to provide native das2 client libraries for other programming environments.

Server Access for New Environments 1: C++ / SDDAS

The ASPERA-3 team at Southwest Research Institute (SwRI) use the C++ based SDDAS (www.sddas.org) toolset for daily tasks while the MARSIS group at U. Iowa employ Autoplot and a custom das2 Java program. Measurements from these two instruments are highly complimentary, so providing immediate access to daily working data sets across both teams is desirable. To prepare for integration, U. Iowa's C-based libdas2 was enhanced with new client capabilities (which were also used for the python module). In turn, SwRI will setup a das2 server to provide access to IDF (Instrument Data Format) data stores.



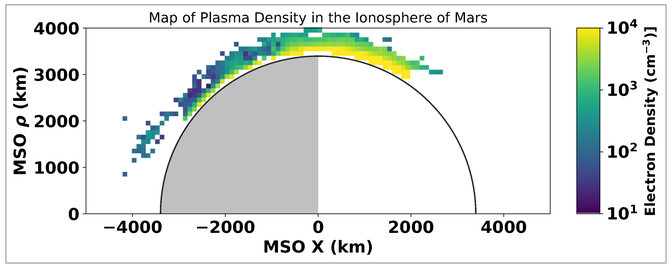


SDDAS display examples above, libdas2 dataset acquisition example below

```
#include <das2/core.h>
DasHttpResp res;
das_http_getBody("http://planet.physics.uiowa.edu/das/das2Server?server=dataset&dataset= Mars_Express/"
                 "MARSIS/Radargram&start_time=2007-001&end_time=2007-002", NULL, NULL, &res);
DasIO* pIn = new_DasIO_socket("TestProgram", res.pSsl, "r");
DasDsBldr* pBldr = new_DasDsBldr();
DasIO_addProcessor(pIn, (StreamHandler*)pBldr);
DasIO_readAll(pIn);  size_t uDatasets;
DasDs** lDs = DasDsBldr_getDataSets(pBldr, &uDatasets);  /* Dataset structures now in lDs pointer array */
```

Server Access for New Environments 2: Python / NumPy

Python is rapidly becoming ubiquitous within the physical sciences. The new das2 python module provides efficient transport of das2 streams directly into Numpy array storage without copies or temporary python heap objects. Additionally meta-data is supplied which categorizes the resulting arrays by their conceptual use as coordinates or data, bin centers, references, offsets, standard deviations, etc. and insures that array dimensions are not conflated with the inherent physical dimensions spanned by the data set. The purpose of the module is to clearly provide higher level plotting packages with the information they need to correctly render array values with a minimum of human intervention.

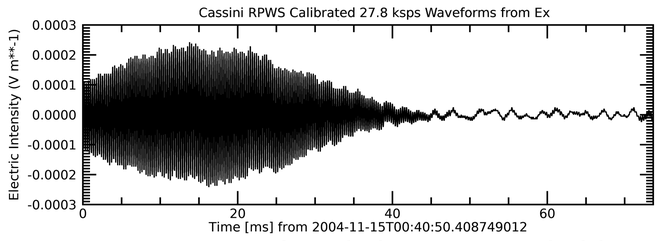


MARSIS plasma densities using matplotlib/das2, courtesy of Zachary Girazian, download ex. below

```
import das2

# Gather local plasma densities
src     = das2.Source("tag:das2.org,2012:site:/uiowa/mars_express/marsis/ne-density/das2")
datasets = src.get( {"time" = ("2005-08-06", "2005-08-07", 1.0) } )
density  = datasets[0]

# Get the numpy arrays
aMSO_X = dataset.coords['X']['center'][ : ]
aMSO_Perp = dataset.coords['rho']['center'][ : ]
aNumberDensity = dataset.data['Ne']['center'][ : ]

# Acquisition is done, rest is standard matplotlib plotting...
```

Server Access for New Environments 3: Pure IDL

Much of the space physics research community runs on IDL (www.harrisgeospatial.com). While Autoplot provides a useful bridge between das2 servers and IDL, a pure IDL module is desirable and could also be folded into larger packages such as SPEDAS (spedas.org). A native IDL das2 module has been written and is undergoing further development.



Cassini RPWS WBR waveform rendered using pure IDL procedure below

```
IDL> .compile das2reader
IDL> .compile das2time

; specify query parameters and get the data
sDataSet = 'Cassini/RPWS/HiRes_MidFreq_Waveform'
sBeg = '2004-11-15T00:40.50.400'
sEnd = '2004-11-15T00:40.50.500'
sParams = [ '10khz', 'Ew=false']
d = das2reader(sDataSet, sBeg, sEnd, params=sParams, /verbose)

; access dataset properties and plot
prop = d[0].stream.properties
p = plot( d[1].ydata, d[1].data, xtit=prop._string_xlabel, ytit=prop._string_ylabel, $
          tit=prop._string_title )
```

Addressing Weaknesses: Cryptic Controls, Static Links

- Many das2 data sources have specialized options. To configure these, scientists currently have to enter "magic strings" in Autoplot or into their download scripts. A small grammar for describing options is needed so that GUIs can be auto-generated.
- Client programs such as Autoplot save the absolute location to a data source in their configuration files. In addition, some specialized das2 clients even store data locations as *compiled-in defaults*. When servers are off line or data are migrated, access fails.

Solution: A Distributed Catalog Using... Webpages?

For sake of simplicity, and to meet typical IT security requirements, static web content is preferred. Four fundamental components: URLs, URIs, and Static XML or JSON pages are sufficient for a robust, distributed catalog system, as long as a few rules are applied.

1) Each page defines a single catalog node.

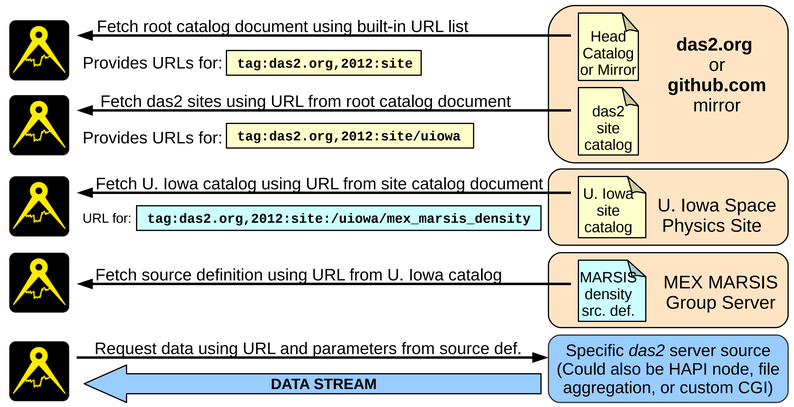
2) Pages referencing other pages must use *fully qualified* URLs.

3) Pages written in a simple limited grammar intended for use by other programs, not humans.

4) Each page must have a UID but for automatic fail-over each should *not* have a unique URL.

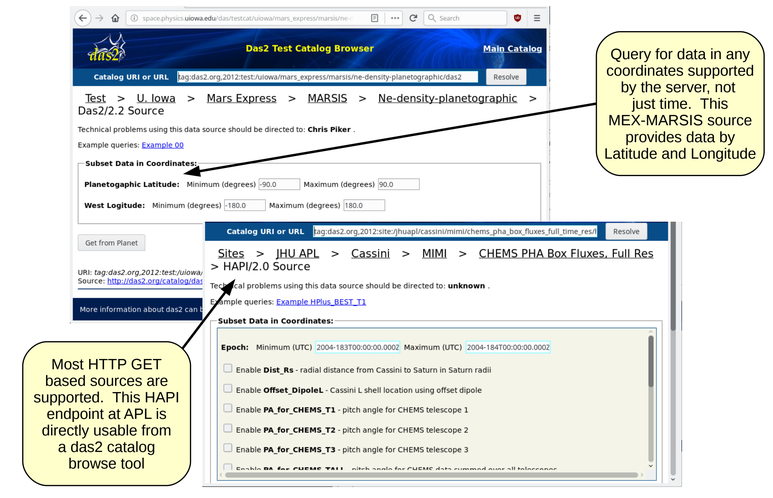
Resolution example: From UIDs to Plots

Instead of saving specific locations (URL) in configuration files, clients save a data source's unique identifier (UID). In the following diagram Autoplot resolves the source UID: **`tag:das2.org,2012:site:/uiowa/mex_marsis_density`** and downloads data.



Working Reference Catalog Client: das2.org/browse

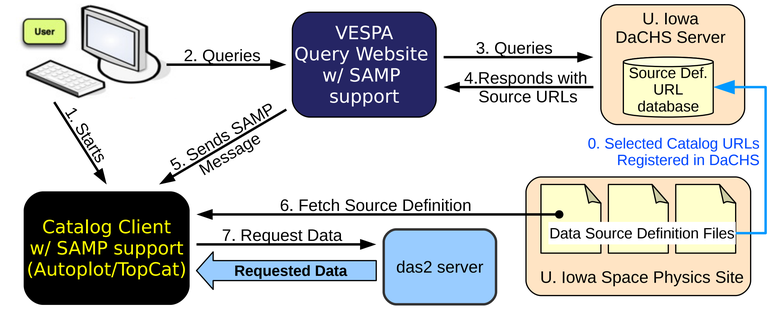
A reference catalog client has been created as a proof of concept and as a developer assistance tool. Currently works with most das2 data sources and any HAPI data node. A small update remains in order to handle the particularly detailed parameters set for Cassini RPWS Survey.



http://das2.org/browse/jhuapl/cassini/mimi/chems_pha_box_fluxes_full_time_res/

Potential Virtual Observatory Integration via DaCHS

We propose that data sources of sufficient reliability and interest to the general research community would be published in-place by registering these within a DaCHS (soft.g-vo.org/dachs) server. The Virtual European Solar and Planetary Access (VESPA) project provides a query interface for such data sources at: www.europlanet-vespa.eu/.



Acknowledgments

Das2 was originally developed in support of the Cassini-Hyygens mission. New features are added as needed in support of NASA/JPL contracts for the Juno, Mars Express, Voyager missions. Travel to the VESPA workshop was provided by the Europlanet 2020 Research Infasturcture programme.

Contacts and Code

To add your site to the das2 catalog contact larry-granroth@uiowa.edu
All das2 source code is freely available under LGPL @ https://saturn.physics.uiowa.edu/svn/das2 and soon @ github.com/das-developers