Open Data Interface - SAAPS/SEDAT/SPENVIS (ODI-SSS) Database Technical Note

Peter Wintoft
Swedish Institute of Space Physics
and
Daniel Heynderickx
DH Consultancy

June 10, 2010

Swedish Institute of Space Physics

DH Consultancy

Document status sheet

Version	Date	Comment
1.0	2009-09-25	First version.
1.1	2010-05-24	Added SEDAT section.
1.2	2010-06-10	Updated SEDAT and SPENVIS sections.

Contents

1 Summary **4**

2 Abbreviations **4**

3 Introduction **4**

4 SAAPS **5**
4.1 SAAPS 1.2 5
4.2 SAAPS 2.0 5
4.2.1 The `saaps.dbt.ODIDatabase` class 5
4.2.2 Implementation 7

5 SEDAT **7**
5.1 Setting up SEDAT with ODI 8
5.2 SEDAT routines modified for use with ODI 8
5.2.1 Perl scripts 8
5.2.2 IDL scripts 9

6 SPENVIS **9**
6.1 Setting up SPENVIS with ODI 10
6.2 New templates and scripts 10
6.2.1 New and updated template files 10
6.2.2 php scripts 11
6.2.3 Other modifications 11

1 Summary

This document describes the adaption of SAAPS, SEDAT, and SPENVIS to operate against the ODI database.

2 Abbreviations

ACE Advanded Composition Explorer

ASCII American Standard Code for Information Interchange

CDF Common Data Format

GOES Geostationary Operational Environment Satellite

HTTP Hypertext Transfer Protocol

ITT/IDL ITT Interactive Data Language

ODI Open Data interface

SAAPS Satellite Anomaly Analysis and Prediction System

SEDAT Space Environment Data Analysis Tool

SOW Statement of Work

SPENVIS Space Environment Information System

SQL Structured Query Language

TSO Time Series Object

3 Introduction

The Open Data Interface (ODI) is a database system that will be accessible and store data from SAAPS, SEDAT, and SPENVIS systems. As ODI is based on MySQL the system is accessible from other systems too. Interfaces to languages like C, Java, PHP, IDL, Matlab exist.

One of the requirements on ODI is that it must be able to store data compliant with the CDF/ISTP/PRBEM guidelines. ISTP defines a set of metadata that must be present to describe solar-terrestrial physics data. ODI is constructed so that it can hold all the ISTP metadata and therefore becomes a general system for storing solar-terrestrial physics data. ODI makes use of the CDF/ISTP metadata definition, seen e.g. in the CDF skeleton files, to populate the ODI data tables holding the metadata. The same approach is also used for non-CDF files, like ASCII files. The PRBEM guidelines

extend the ISTP guidelines by defining a set of variables relevant for radiation belt data. The PRBEM extension does not affect the ODI database model.

In the following sections we describe the underlying ISTP metadata, how it is stored in ODI, how tables to hold metadata and data are created, how data are ingested and exported.

4 SAAPS

The latest version of SAAPS that works against the SAAPS database is 1.2. In this work SAAPS 1.2 is developed to work against the ODI database. This version is called SAAPS 2.0.

SAAPS 1.2 was a complete system consisting of both tools and database. The database was updated with real time data. In SAAPS 2.0 only the tools are needed as the ODI system keeps the database up to date.

4.1 SAAPS 1.2

SAAPS is implemented in Java and is organised into three basic packages according to Table 1, and a few additional support packages.

Table 1: The main packages in SAAPS.

Package name	Description
`saaps.dbt`	The database tools to access data from the database.
`saaps.saam`	The satellite anomaly analysis module.
`saaps.sapm`	The satellite anomaly prediction module.

The entry point to access the database in SAAPS 1.2 is the class `saaps.dbt.Database`. This class provides methods to connect to the database, read and write data, and read and write metadata.

4.2 SAAPS 2.0

In SAAPS 2.0 the `saaps.dbt.Database` class is replaced by `saaps.dbt.ODIDatabase` that keeps the same functionality for reading but instead connects to the ODI database. The writing methods are not implemented as they are not used. All classes that previously used `saaps.dbt.Database` are updated to use `saaps.dbt.ODIDatabase` and they are listed in Table 2.

4.2.1 The `saaps.dbt.ODIDatabase` class

The `saaps.dbt.ODIDatabase` class provides methods for connecting and reading data from the ODI database. A connection is made according to

Table 2: All classes that are updated to use `saaps.dbt.ODIDatabase`.

Class	Description
`saaps.dbt.DatabaseItems`	A class to hold the names of the datasets.
`saaps.dbt.ShowMetaData`	A class to show the metadata relevant for SAAPS.
`saaps.saam.DataPlotterApp`	Tool to plot data from the ODI datasets.
`saaps.saam.ElFluence`	A class used to extract daily electron fluences from GOES. Used by the Electron Fluence Levels tool in the Analysis Module.
`saaps.saam.SEAApp`	A tool to perform Superposed Epoch Analysis using data from the ODI database.
`saaps.saam.SumKp`	A class to extract the daily sums of Kp over a period. Used by the Sum Kp Level tool of the Analysis module.
`saaps.sapm.Anom001Model`	A model for predicting anomalies for satellite type 001.
`saaps.sapm.Anom002Model`	A model for predicting anomalies for satellite type 002.
`saaps.sapm.Anom003Model`	A model for predicting anomalies for satellite type 003.
`saaps.sapm.Anom004Model`	A model for predicting anomalies for satellite type 004.
`saaps.sapm.ElFluence`	A class to compute the daily electron fluence from GOES.

```
ODIDatabase db = new ODIDatabase()
```

which will connect to the database at user level 3 based on the ODI environment variables. See TN ODI.

The basic method to read data is

```
TimeSeries ts = db.retrieveTimeSeries(String dataset, Date start, Date end)
```

where `TimeSeries` is a SAAPS class to handle time series data.

4.2.2 Implementation

For SAAPS to function the following environment variables needs to be set:

```
$SAAPS_DIR  : The directory where SAAPS is installed.
$SAAPS_HOME : The directory where the SAAPS executables are installed.
$SAAPS_LIB  : The directory of external packages.
```

On `spitfire.estec.esa.int` these have been set to

```
SAAPS_DIR=$ODI_HOME/SAAPS
SAAPS_HOME=$SAAPS_DIR
SAAPS_LIB=$SAAPS_DIR/lib
```

To run SAAPS the following is executed on the command line

```
$SAAPS_HOME/saaps.sh
```

which will start the SAAPS user interface.

5 SEDAT

SEDAT uses the STPDF as an engine to manage and access its datasets. STPDF keeps track of the available datasets through a series of control and descriptive files (tin and chp files). SEDAT distinguished between system datasets and user datasets: the former are ingested and controlled by the system administrator, the latter are created and managed by individual users.

In the ODI project, the STPDF backend to the system datasets has been replaced by a MySQL backend. Handling of the user datasets has not been modified, which means that the upgraded version of SEDAT uses ODI as well as STPDF.

The modifications made to the SEDAT system only affect the backend for system datasets, and are transparent to users using the SEDAT GUI. Thus, users will not see any changes in the operation of SEDAT.

5.1 Setting up SEDAT with ODI

In order to set up SEDAT with ODI, an ODI server needs to be set up and visible to the SEDAT server. Only read privileges are needed for the ODI server queries made by SEDAT, as system datasets are not modified directly by SEDAT (but rather are managed through external processes like the ODI cron job data ingestion tools), and user datasets will still be managed by STPDF.

The IDL scripts used to read data from system datasets have been modified to use SQL queries to ODI instead. IDL performs SQL queries through ODBC, which requires an IDL Dataminer licence. Setting up the ODBC connection for IDL is explained in the IDL Dataminer user manual. It mainly consists of creating an .odbc.ini file in the SEDAT system root directory.

5.2 SEDAT routines modified for use with ODI

The SEDAT GUI interacts with the SEDAT system through a series of Perl scripts, which in turn spawn STPDF applications or IDL script runs.

5.2.1 Perl scripts

Only three Perl scripts needed to be modified in the SEDAT `server/cgi-client` directory:

- `02-D-Li.pl`: outputs (part of) a dataset in ASCII or CDF format;
- `02-D-Op.pl`: returns a description of a dataset, the variables in the dataset and the number of records;
- `02-D-PS.pl`: searches for system datasets by property (region, particle type, energy range).

The modifications were made such that processing of user datasets was not affected, and the functionality of system dataset processing is unchanged from the standpoint of the GUI.

In addition, `Sedat.pm.in` was extended with a subroutine to open a connection to the ODI server. The connection parameters were added to `server/param.in`: the database user name and password, the server host name and socket and port need to be specified here (the names of the 6 parameters are prefixed with `ODI_`.

In the original SEDAT, a administrator script `server/admin/addds.pl` is run when new datasets are added. This script creates subdirectories and information files for each dataset. A new script `odi_addds.pl` was added to process system datasets: this should be run each time a new dataset is added to the ODI server which needs to be visible to SEDAT.

Finally, the `server/server/sedat_build_query.pl` script was extended with an `IDLexecute(".COMPILE odi_tools");` command to ensure compilation of the IDL ODI tools (see next section).

5.2.2 IDL scripts

The SEDAT IDL scripts communicate with STPDF through external calls to the STPDF C library routines. This mechanism had to be replaced by Dataminer calls to the ODI MySQL database.

The SEDAT IDL system scripts reside in `/user/users/SYSTEM/idl`. A library of ODI IDL tools was written and collected in `odi_tools.pro`. One of the library procedures (`DBConnect`) opens a connection to the ODI database server through IDL Dataminer. This procedure uses the environment variable `ODI_SEDAT_ODBC`) which contains the name of the ODBC resource defined in the `.odbc.ini` file (see Section 5.1). If this environment variable is not set, the name defaults to `ODI_SEDAT`.

Two procedures, `odi_ascii.pro` and `odi_cdf.pro`, were written to output (parts of) a dataset to ASCII or CDF files, respectively. These procedures are called by the `server/cgi-client/02-D-Li.pl` script.

The remainder of this section treats the modifications made to the IDL dataset open and read scripts. As for the cgi-client Perl scripts, only the handling of system datasets was modified, in a way which is transparent to the procedures that call these routines. The following scripts were modified:

- `sedat_make_dmeta.pro`: two tags were added to the `dmeta` structure to hold the database connector unit and the SQL query for reading the specified variables.
- `sedat_open_read.pro`: when a system dataset needs to be opened, the new routine `sedat_open_read_odi.pro` is called; this routine sets up the necessary metadata and also constructs the SQL query string for reading the data.
- `sedat_read_record.pro`: when a system dataset needs to be read, the new routine `sedat_read_record_odi.pro` is called.
- `sedat_close.pro`: the database connection is closed, and the corresponding object deleted.

6 SPENVIS

SPENVIS uses a custom built set of IDL tools to:

- create cdf files from downloaded datasets;
- produce HTML templates for each dataset;
- query the cdf files;
- plot the results from the queries.

In the ODI project, this data backend has been replaced by a MySQL backend. To this effect, new HTML templates and php scripts have been written that query ODI and display dataset information without the need for dataset specific templates.

6.1 Setting up SPENVIS with ODI

In order to set up SPENVIS with ODI, an ODI server needs to be set up and visible to the SPENVIS server. Only read privileges are needed for the ODI server queries made by SPENVIS. All database queries are performed by php scripts, so it is not necessary to install an IDL Dataminer licence or set up an ODBC for IDL.

The template files in `templates/databases` directory are no longer needed, in fact this subdirectory can be deleted entirely.

6.2 New templates and scripts

The ODI implementation in SPENVIS uses a combination of php scripts and HTML templates. The navigation buttons in the templates contain code to run the respective php script prior to navigating to another template. The outputs of the php scripts contain HTML code which is ingested in the display of the target template. This procedure provides for a dynamic display of database information using a combination of fixed templates and content generated on the fly.

6.2.1 New and updated template files

All new and updated template files are located in the `templates` subdirectory. The following files have been updated:

- `modtable.html, modtable_jupiter.html, modtable_mars.html`: the link to the main query page has been changed.
- `results_earth.html, results_jupiter.html, results_mars.html`: the links to the query pages have been changed, as well as the list of output files.

The following files have been added:

- `db_main.html`: the start template which displays a list of datasets; this templates uses output from the php script `dblist.php`.
- `db_channels.html`: this template displays a list of variables for the dataset selected on the main page; this templates uses output from the php script `dbchannels.php`.
- `db_out.html`: the output template displays links to the output files (text and graphics), and calls the IDL plotting tool `db_plot.pro`; this templates uses output from the php script `dbfile.php`.

In addition to the templates in the `templates directory`, a number of help pages in the `htdocs/help/models` subdirectory were modified: `databases.html, packages.html, packages_jupiter.html, packages_mars.html` and all HTML files in the `databases` subdirectory; from this subdirectory, the file `databasesmain.html` has been removed.

6.2.2 php scripts

Four php scripts have been added to the `bin` directory:

1. `dblist.php`: this script queries the database and produces a list of datasets with start and stop times in HTML table format; the result is written to the output file `spenvis_dbl.txt` which is read in when the template `db_main.html` is displayed.

2. `dbchannels.php`: this script queries the database and produces a list of variables for the selected dataset in HTML format, with check boxes for selections; the result is written to the output file `spenvis_dbc.txt` which is read in when the template `db_channels.html` is displayed.

3. `dbfile.php`: this script queries the database and retrieves the data records of the selected dataset and variables; the resulting records are written to the output file `spenvis_dbo.txt` with the standard SPENVIS header information. Epoch is always output, in AMJD format. A second output file `spenvis_dbf.txt` contains HTML code for variable selection menu on the output page `db_out.html`.

4. `dbtools.php`: this file contains a number of functions that are called by the main php scripts.

6.2.3 Other modifications

In the `bin` directory, the control files `mapping.txt` and `refresh.txt` were updated to reflect the new implementation.

In the `idl` subdirectory, a new plot procedure `db_plot.pro` has been added. The following files were removed:

```
cdf_meta.pro
dbplot.pro
dbplot_batch.pro
merge.pro
mission_sel.pro
mission_sel1_batch.pro
mission_sel_batch.pro
query.pro
query_batch.pro
```

In the `htbin` subdirectory, a new file `ODI.properties` was added. This file contains the connection parameters for the ODI database and should be modified to reflect the local ODI setup.