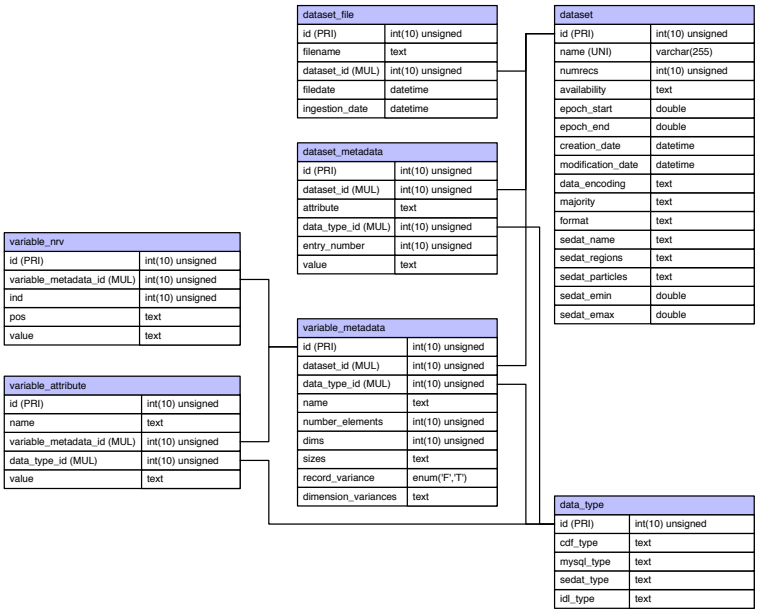
ODI User Guide
Version 1.1

March 30, 2010



Peter Wintoft, Swedish Institute of Space Physics

Daniel Heynderickx, DH Consultancy

Document status sheet

Version	Date	Comment
1.0	2009-09-25	First version.
1.1	2010-03-30	Updated.

Contents

1 Summary

The Open Data Interface (ODI) was developed to store space physics data and metadata compliant with the CDF/ISTP/PRBEM guide lines. The ODI system shall also be capable to ingest data from various sources, such as ASCII files, HTML pages, and CDF files. As a CDF file contains a complete description of metadata, and as the ISTP guide lines builds on CDF, is is natural to use the CDF metadata syntax in ODI. However, the data is not stored in files in ODI but rather in a database and it was chosen to use MySQL.

In this guide we describe how to explore the database and pull data out of the system.

2 Prerequisites

The ODI system must have been properly installed and set up together with all necessary software such as PHP and MySQL. This is described in the *ODI Administrator Guide*. It is assumed that ODI is installed under `$ODI_HOME`.

In addition to `$ODI_HOME` a few other environment variables must be set. To check the variables do

```
>> cd $ODI_HOME/tools
>> check_odi.php
```

where `>>` is the Unix shell prompt.

3 Overview

The ODI system builds on a MySQL database. For an end user it is only possible to read data and metadata from the system, it is not possible to modify the database. As the ODI database is a MySQL database the data can be directly accessed in many different ways, e.g. from the `mysql` command line, PHP, Java, IDL, Matlab, and so on.

ODI contains two categories of tables: tables that hold *metadata* and tables holding *data*. The metadata tables are shown in Appendix A. These tables contain all metadata for all datasets stored in ODI. Each table contains a unique key (`id`) that is used to tie e.g. the metadata to a specific variable, for a specific dataset. In this way queries can be constructed to find the right metadata.

To each dataset there is an associated table that holds the data. The names of the datatables always start with `dataset_` followed by a unique name following the syntax `dataset_<platform>_<instrument>` like `dataset_xmm_rm`. The data are organised in the tables with the following column names:

```
cdf_epoch, epoch, millisec, <var 1>, <var 2>, ..., dataset_file_id
```

where `cdf_epoch` is the CDF epoch, `epoch` is the epoch in `YYYY-MM-DD hh:mm:ss` format, `millisec` is the millisecond part. Then follows the variables, one column for each variable, and finally there is a `dataset_file_id` column that is used internally by the ODI system to keep track of ingested datasets.

The stored data may either by scalar or multidimensional variables. As an SQL database can not store multidimensional variables these variables need to be processed in some way. ODI solves this by expanding a multidimensional variable over several columns in the data table. To give an example, assume we have a one dimensional variable called `position` with 3 elements. The corresponding column names in the table would then be

```
position_1, position_2, position_3
```

When the tools described below are used to access the data this mapping is done automatically, thus, if the `position` variable is requested all three elements will be returned. However, if the data table is accessed directly then the user must keep track of the mapping.

4 ODI tools

4.1 Tools in `$ODI_HOME/tools/`

In `$ODI_HOME/tools/` there are several command line scripts. Here we only describe those that are relevant for an end user.

`check_epoch.php`

> Check if the epochs in a dataset are equidistant. The program determines the median difference between consecutive epoch values and then lists all epochs which have a epoch differences that are different from the median.

`export_build_xml.php`

> Create an XML template file for a dataset that describes the variables and epoch range to be exported.

`export_to_xml.php`

> Export a dataset to a text file in XML format. The variables to be exported and the epoch range is specified in the export template file created using the previous program. The export template file may first be edited to reflect the chosen variables and epoch range.

`show_datasets.php`

> Get a list of the ODI datasets.

`show_metadata.php`

Show the metadata for a selected dataset.

`show_updates.php`

List all datasets that have been updated for a given epoch range.

`show_variables.php`

Show the variables for a selected dataset. It lists first the non-record variant variables and then the record variant variables.

4.2 Exporting data to text file

To export a dataset the epoch range and variables must be specified. This information is given in an XML settings file. A template settings file can be generated with the command

```
export_build_xml.php <dataset>
```

which will create the file `<dataset>.set.xml`. This file may then be edited. Here one can change the name of the output file, the epoch range, what variables to export and the order of the variables. They are given within the XML tags `<export_file></export_file>`, `<epoch_start></epoch_start>`, `<epoch_end></epoch_end>`, and `<variable></variable>`, respectively. The next step is to run

```
export_to_xml.php <settings file>
```

which will produce the export file, which is also an XML file. The file contains a header section with the field names and a data section with each field separated by commas.

4.3 Exporting data to CDF file

The ODI data and metadata can be exported to a CDF file using the IDL program

```
odicdf, cdf_name, dataset_name, varnames=varnames, $   \\
        epochrange=epochrange, recordrange=recordrange
```

where the name for the cdf file, the SEDAT dataset name, the variables to be extracted (all variables if this keyword is not specified), and epoch or record range is given (one and only one of epoch or record range).

4.4 Reading data using PHP

A PHP function is defined in the ODI library code to read data from a dataset. The variables and the epoch range are given to the functions argument list and an array with the data is returned. The syntax is

> array `odi_read_data` (string `$dataset` , string `$epoch_start` , string `$epoch_end` , string `$variable_1` [, `$variable_2` [, ...]])

where

- `$dataset` is the SEDAT dataset name, e.g. `GOES_I11_5`,
- `$epoch_start` is the start date with the format `"YYYY-MM-DD hh:mm:ss.msec"`, e.g. "2010-03-29 12:40:20.155",
- `$epoch_end` is the end date,
- `variable_1` is the name of a variable, followed by optionally more variables.

The code below shows a PHP program that reads some GOES/SEM data and displays it. The code can be found at `$ODI_HOME/test/odi_read_data_test.php`.

```
#!/usr/bin/php
<?php

require_once($_SERVER["ODI_HOME"].'/lib/odi.library.php');

$dataset = "GOES_I11_5";
$epoch_start = "2010-01-01 00:00:00.000";
$epoch_end = "2010-01-01 00:59:59.999";

$x = odi_read_data($dataset,$epoch_start,$epoch_end,"epoch","position","fpio");

print_r($x);

?>
```

Running the program produces the following output:

```
Array
(
    [0] => Array
        (
            [Epoch] => 2010-01-01 00:00:00.000
            [Position (X)] => -30130.1
            [Position (Y)] => 29505.6
            [Position (Z)] => 0
```

```
            [FPIO (> 1 MeV proton flux (corrected))] => 5.13
            [FPIO (> 5 MeV proton flux (corrected))] => 0.292
            [FPIO (> 10 MeV proton flux (corrected))] => 0.255
            [FPIO (> 30 MeV proton flux (corrected))] => 0.193
            [FPIO (> 50 MeV proton flux (corrected))] => 0.18
            [FPIO (> 60 MeV proton flux (corrected))] => 0.177
            [FPIO (> 100 MeV proton flux (corrected))] => 0.103
        )

    [1] => Array
        (
            [Epoch] => 2010-01-01 00:05:00.000
            [Position (X)] => -30130.1
            [Position (Y)] => 29505.6
            [Position (Z)] => 0
            [FPIO (> 1 MeV proton flux (corrected))] => 9.77
            [FPIO (> 5 MeV proton flux (corrected))] => 0.418
            [FPIO (> 10 MeV proton flux (corrected))] => 0.219
            [FPIO (> 30 MeV proton flux (corrected))] => 0.148
            [FPIO (> 50 MeV proton flux (corrected))] => 0.071
            [FPIO (> 60 MeV proton flux (corrected))] => 0.0578
            [FPIO (> 100 MeV proton flux (corrected))] => 0.0308
        )
```

where only the first two records are shown.

4.5 Reading data using IDL

4.6 Reading data using Java

To read data from ODI in a Java program the `readDataset` method in the class `odi.Database` can be used. The syntax is

> *Dataset* `readDataset` (*String* `datasetName` , *String* `epochStart` , *String* `epochEnd` , *String[]* `variables`)

where *Dataset* is a class holding the metadata and data.

The program listed below illustrates how to read data from ODI. The code can be found at `$ODI_HOME/test/OdiReadDataTest.java`.

```
import odi.*;

public class OdiReadDataTest {

    public static void main(String[] args) {
```

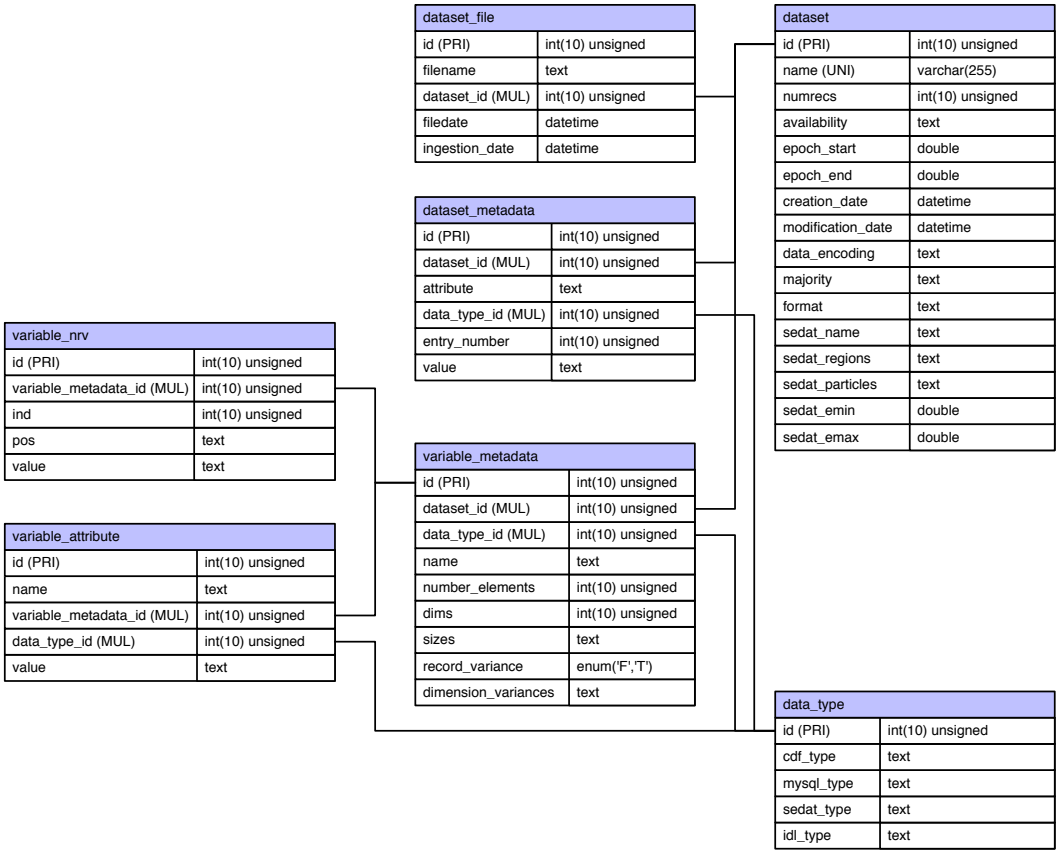
```
        String dataset = "goes_i12_5";
        String epochStart = "2010-01-01 00:00:00.000";
        String epochEnd = "2010-01-01 00:59:59.999";
        String[] variables = {"epoch","position","fpio"};

        try {
            Database db = new Database();
            Dataset ds = db.readDataset(dataset,
                                        epochStart, epochEnd,
                                        variables);
            System.out.println(ds);
        }
        catch(DatabaseException e) {
            System.err.println(e);
        }

    }

}
```

A Metadata


dataset_file
id (PRI) int(10) unsigned
filename text
dataset_id (MUL) int(10) unsigned
filedate datetime
ingestion_date datetime
dataset
id (PRI) int(10) unsigned
name (UNI) varchar(255)
numrecs int(10) unsigned
availability text
epoch_start double
epoch_end double
creation_date datetime
modification_date datetime
data_encoding text
majority text
format text
sedat_name text
sedat_regions text
sedat_particles text
sedat_emin double
sedat_emax double
dataset_metadata
id (PRI) int(10) unsigned
dataset_id (MUL) int(10) unsigned
attribute text
data_type_id (MUL) int(10) unsigned
entry_number int(10) unsigned
value text
variable_nrv
id (PRI) int(10) unsigned
variable_metadata_id (MUL) int(10) unsigned
ind int(10) unsigned
pos text
value text
variable_metadata
id (PRI) int(10) unsigned
dataset_id (MUL) int(10) unsigned
data_type_id (MUL) int(10) unsigned
name text
number_elements int(10) unsigned
dims int(10) unsigned
sizes text
record_variance enum('F','T')
dimension_variances text
variable_attribute
id (PRI) int(10) unsigned
name text
variable_metadata_id (MUL) int(10) unsigned
data_type_id (MUL) int(10) unsigned
value text
data_type
id (PRI) int(10) unsigned
cdf_type text
mysql_type text
sedat_type text
idl_type text

dataset	**A table holding the names of each dataset_* table together with some key information.**
id	A unique identifier.
name	The name of the dataset table in the ODI database.
numrecs	The number of records in the data table.
availability	A flag indicating whether the dataset is public or private.
epoch_start	The first epoch in the dataset.
epoch_end	The last epoch in the dataset.
creation_date	The date when the dataset_* table was created.
modification_date	The last date when the dataset was modified.
data_encoding	The CDF header attribute DATA ENCODING.
majority	The CDF header attribute MAJORITY.
format	The CDF header attribute FORMAT.
sedat_name	SEDAT datset name without SYSTEM! prefix.
sedat_regions	SEDAT region code.
sedat_particles	SEDAT particle code.
sedat_emin	SEDAT min. energy (MeV).
sedat_emax	SEDAT max. energy (MeV).

dataset_file	**A table to hold the file names of all ingested data files.**
id	A unique identifier.
filename	The file name of the ingested file.
dataset_id	A key to the associated dataset in table dataset.
filedate	The time stamp of the raw data file.
ingestion_date	The date when the file was ingested into ODI.

dataset_metadata	**Metadata for each dataset.**
id	A unique identifier.
dataset_id	A key to the associated dataset in table dataset.
attribute	The attribute name. This corresponds to the CDF global attribute.
data_type_id	A key to the associated data type in table data_type.
entry_number	The CDF global parameter entry number.
value	The value (or contents) of the dataset attribute.

data_type	**CDF data types together with associated MySQL and SEDAT data types.**
id	A unique identifier.
cdf_type	The CDF data type.
mysql_type	The MySQL data type.
sedat_type	The SEDAT data type.
idl_type	The IDL data type.

variable_attribute	**The attributes for each variable.**
id	A unique identifier.
name	The name of the variable attribute.
variable_metadata_id	A key to the associated variable in table variable_metadata.
data_type_id	A key to the associated data type in table data_type.
value	The value (or contents) of the variable attribute.

variable_metadata	**The metadata for each variable.**
id	A unique identifier.
dataset_id	A key to the associated dataset in table dataset.
data_type_id	A key to the associated data type in table data_type.
name	The name of the variable.
number_elements	The CDF variable parameter Number Elements.
dims	The CDF variable parameter Dims.
sizes	The CDF variable parameter Sizes.
record_variance	The CDF variable parameter Record Variance.
dimension_variances	The CDF variable parameter Dimension Variances.

variable_nrv	**The values of the non-record-variant data.**
id	A unique identifier.
variable_metadata_id	A key to the associated variable in table variable_metadata.
ind	An index to the nrv variable. It goes from 1 to the product of the number of elements in each dimension.
pos	The index to the nrv variable as given in the CDF.
value	The value of the nrv variable.