OdimH5 User's Guide

Software version: 2.14a
Last update of this manual: April 28, 2011
Authors: Maciej Szewczykowski, Łukasz Wojtas

This document is a user's guide on how to use OdimH5.

OdimH5 is a console HDF5 exchange software designed to work on radar data files. It provides XML descriptor handler, HDF5 converter, FTP and Baltrad feeder mode.

1 Introduction

1.1 License

1.2 Overview

The OdimH5 is a Java-based tool working on meteorological radar data. It allows users to create an XML descriptor which contains all major information require to create a HDF5 file. The application also allows users to convert specific radar data to HDF5 format based on the descriptor. For `Baltrad` users it provides automatic online mode which feeds `BaltradDex` with actual data.

The application was implemented using the Java™ 2 Platform, which is machine-independent.

HDF libraries

This release was built and tested with HDF5-1.8.4 Patch 1 with HDF5 1.6 compatibility flag. For information on new features in HDF5 Release 1.8.0 and

format compatibility considerations, please visit
http://www.hdfgroup.org/HDF5/doc/ADGuide/CompatFormat180.html.

Apache libraires

This software was built using Apache Commons libraries http://commons.apache.org/.

jpathwatch library

This software was built using jpathwatch 0.94 http://jpathwatch.wordpress.com/.

Platforms

This release was built and tested for the following platforms:

- Linux x86_64
- 32-bit Windows XP/Vista/7

1.3 Supported radar systems

This version can work with limited radar systems and products listed below.

Platforms

- Gematronik RAINBOW

Type of product

- Polar Volume Scan (ver. 5.2x and 5.3x)
  - dBZ - Reflectivity
  - uPhiDP - Differential Phase Shift
  - KDP - Specific Differential Phase Shift
  - RhoHV - Correlation Coefficient
- Cartesian image and composite (ver. 5.2x)
  - PPI - Plan Position Indicator
  - CAPPI - Constant Altitude PPI
  - MAX - Maximum Display
  - EHT - Echo Height
  - SRI - Surface Rainfall Intensity
  - PAC - Precipitation Accumulation
  - VIL - Vertical Integrated Liquid
  - HSHEAR - Horizontal Shear

- Range-height indicator
  - RHI - Range Height Indicator

1.4 Features

- Convert a single file to HDF5.
- Convert a file from HDF5 to Rainbow format.
- Convert and send files via FTP in continueus mode.
- Convert and send files to Baltrad Dex in continueus mode.

2 Getting Started

2.1 Installation

To get newest version of OdimH5 use `Git` a distributed revision control system and clone project from baltrad server. To do this use following command

```
git clone gitosis@git.baltrad.eu:OdimH5.git
```

or download it from Opera FTP server (`ftppro.knmi.nl`) and extract.

After downloading use `Apache Ant` to compile sources.

```
ant -Dprefix=/my/install/dir install
```

Program will be installed in `/my/install/dir`. If folder path parameter is not provided Odim will be installed in default folder `/opt/OdimH5`. Program uses HDF5 libraries, which are mostly included to the `.jar` file. However JNI interfaces files need separate installation.

In Linux: `libjhdf.so` and `libjhdf5.so` files which are provided with main program (`lib/linux`) must be included to the `LD_LIBRARY_PATH`.

In Windows: `jhdf.dll` and `jhdf5.dll` files which are provided with main program (`lib/win`) must be included to system environment variables.

2.2 Settings

The program reads options from `options.xml` file stored in the main folder. The following options can be provided:

`<radar>`	Every radar is represented by this element. The attribute `name` should be 3-letter radar name, same as one stored in raw volume file e.g. `name="SWI"`. This section contains one obligated tag: `WMO_id`, and following extra tags: `file_name`, `directory` and `nrays`

<WMO_id>	WMO block and station number
<file_name>	File name prefix compliant with ODIM file naming convention: pflag_productidentifier_oflag_originator_. For details see Opera document WD-2010-02.
<directory>	Path of the directory where radar volume data are stored. This directory will be watched in feeder mode by application, and every new file in the directory will awake the process of conversion.
<nrays>	Number of rays that are allowed, if volume contains more then that number, extra rays are discard.
<location>	Full radar name for BALTRAD DEX. If not provided Odim will use name from RAINBOW volume file.
<baltrad>	Section containing BALTRAD options for server and sender.
<server>	Address of HTTP server.
<sender>	Sender name.
<ftp>	Section containing FTP options for address, login and password.
<radars>	List of radars to be sent using FTP. Use 3-letter names separeted by space.
<address>	Address of FTP server.
<login>	Login to FTP server.
<password>	Password to FTP server.

Example options.xml file:

```
<?xml version="1.0" ?>
<!-- options -->
<options>
   <radar name="BRZ">
      <WMO_id>12568</WMO_id>
      <file_name>T_PAGZ46_C_SOWR_</file_name>
      <directory>/home/volumes/BRZ_250_Z.vol</directory>
      <nrays>360</nrays>
      <location>full_name</location>
   </radar>
   <baltrad>
      <server>http://172.30.9.34:8084/BaltradDex/dispatch.htm</server>
      <sender>Baltrad.IMGW.pl</sender>
   </baltrad>
```

```
   <ftp>
      <radars>BRZ POZ LEG</address>
      <address>ftp.address</address>
      <login>login</login>
      <password>pass</password>
   </ftp>
</options>
```

2.3 Convertion mode

`OdimH5` provides two conversion modes. First one creates XML descriptor, that can be used later to create HDF file. Second mode converts raw data directly to HDF5.

Prepare descriptor

Descriptor is an XML file, which structure corresponds to HDF5 file. To prepare descriptor use the following parameters:

`-i` Input file's path.
Program can work with only one file simultaneously.

`-o` Output file's path.
It is suggested to use `.xml` filename extension.

`-p` Radar platform's name.
At the moment only Gematronik's RAINBOW software is supported.

`-f` Product format.
Use one of the formats listed above according to input data type.

`-v` Verbose mode.
It is optional and displays status of progress of program work.

Example of use:

```
java -jar OdimH5.jar -i input.ppi -o ppi.xml -p RAINBOW -f IMAGE -v
```

Prepare HDF5 file from descriptor

It requires XML descriptor as an input file. To prepare HDF5 use the following parameters:

`-i` Input file's path.
Program can work with only one file simultaneously.

`-o` Output file's path.
It is suggested to use `.h5` filename extension. If no output file name provided program will generate one using ODIM convention, or if no prefix in options.xml were set program will use input file name with `.h5` filename extension.

`-v` Verbose mode.
It is optional and displays status of progress of program work.

Example of use:

```
java -jar OdimH5.jar -i ppi.xml -o output.h5
```

Example of descriptor file

Header:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--ODIM_H5 descriptor file, platform:  RAINBOW, file object:
PVOL-->
```

Root group:

```
<group name="/">
```

What group:

```
<group name="what">
   <attribute class="string" name="object">PVOL</attribute>
   <attribute class="string" name="version">H5rad 2.0</attribute>
   <attribute class="string" name="date">20101022</attribute>
   <attribute class="string" name="time">123044</attribute>
   <attribute class="string" name="source">WMO:12374</attribute>
</group>
```

Where group:

```
<group name="where">
   <attribute class="double" name="lon">20.960630</attribute>
   <attribute class="double" name="lat">52.405220</attribute>
   <attribute class="double" name="height">119.000000</attribute>
</group>
```

How group:

```
<group name="how">
   <attribute class="long" name="startepochs">1287743444000</attribute>
   <attribute class="long" name="endepochs">1287743444000</attribute>
   <attribute class="string" name="system">GEMA</attribute>
   <attribute class="string" name="software">RAINBOW</attribute>
   <attribute class="string" name="sw_version">5.29.5</attribute>
   <attribute class="double" name="beamwidth">1</attribute>
   <attribute class="double" name="wavelength">0.0531</attribute>
</group>
```

Dataset group:

```
<group name="dataset1">
```

Dataset children:

```
<group name="what">
   <attribute class="string" name="product">SCAN</attribute>
   <attribute class="string" name="startdate">20101022</attribute>
   <attribute class="string" name="starttime">123044</attribute>
   <attribute class="string" name="enddate">20101022</attribute>
   <attribute class="string" name="endtime">123059</attribute>
</group>
<group name="where">
   <attribute class="double" name="elangle">0.5</attribute>
   <attribute class="long" name="nbins">250</attribute>
   <attribute class="double" name="rstart">0</attribute>
   <attribute class="double" name="rscale">1000.0</attribute>
   <attribute class="long" name="nrays">360</attribute>
   <attribute class="long" name="a1gate">0</attribute>
</group>
<group name="data1">
   <group name="what">
      <attribute class="string" name="quantity">DBZH</attribute>
      <attribute class="double" name="gain">0.5</attribute>
      <attribute class="double" name="offset">-32.0</attribute>
      <attribute class="double" name="nodata">255.0</attribute>
      <attribute class="double" name="undetect">0.0</attribute>
   </group>
   <dataset CLASS="IMAGE" IMAGE_VERSION="1.2" chunk="20x20"
   data_size="8" data_type="integer" dimensions="360x250"
   gzip_level="6" name="data">data/data1.dat</dataset>
   </group>
</group>
```

Prepare HDF5 file directly from raw file

It has the same parameters as descriptor preparation mode, but output file name has to end with `.h5`. For volumes `-o` parameter can be skipped, application will convert file directly to HDF5 using default name, which will be ODIM name when proper parameter will be provided in `options.xml` file or standard date-format name otherwise.

Examples of use:

```
java -jar OdimH5.jar -i input.ppi -o ppi.h5 -p RAINBOW -f IMAGE

java -jar OdimH5.jar -i input.vol -p RAINBOW -f PVOL
```

Prepare Rainbow volume file from HDF5 file

Input filename has to have a `.h5` or `.hdf` extension, and output filename has to have `.vol` extension.

For example:

```
java -jar OdimH5.jar -i input.h5 -o output.vol -p RAINBOW -f PVOL
```

2.4 Baltrad Feeder

`OdimH5` allows users to send HDF5 files into `BaltradDex` system. To send a file use following command:

```
java -jar OdimH5.jar -i input.h5 -r Brzuchania -s IMGW.pl
-a http://172.30.9.34:8084/BaltradDex/dispatch.htm
```

It sends a single HDF5 file to the server but it can work as a continuous `Baltrad` feeder as well with online conversion to HDF5 format. It works automatically with specific options provided by user.

To run feeder use `-c` option and provide `Baltrad` details in `options.xml` in `<baltrad>` section.

Example of use:

```
java -jar OdimH5.jar -c
```

```
<?xml version="1.0" ?>
<!-- ODIMH5 options -->
<options>
   <radar name="SWI">
      <WMO_id>12220</WMO_id>
      <file_name>T_PAGZ46_C_SOWR_</file_name>
      <directory>rb5input/SWI</directory>
      <location>Świdwin</location>
   </radar>
   <baltrad>
      <server>http://172.31.50.150:8084/BaltradDex/dispatch.htm</server>
      <sender>rainbow.imgw.pl</sender>
   </baltrad>
</options>
```

2.5 Sending file by FTP

`OdimH5` allows users to send HDF5 file using FTP. It works similar to `Baltrad` feeder. To run FTP feeder use `-c` option and provide `FTP` details in `options.xml` in `<ftp>` section.

```
   <ftp>
      <radars>SWI POZ</radars>
      <address>ftpops.metoffice.gov.uk</address>
      <login>***</login>
      <password>***</password>
      <directory></directory>
   </ftp>
```

2.6 Help

To display help menu in program use following parameter:

```
java -jar OdimH5.jar -h
```

3 Development support and guidelines

This section of document provides information of particular interest to developers of OdimH5 application.

To use OdimH5 API for creating HDF5 file the following classes have to be imported:

```
import ncsa.hdf.hdf5lib.HDF5Constants;
import pl.imgw.odimH5.model.HDF5Model;
```

3.1 Creating new HDF5 file

To create new file use **H5Fcreate_wrap** method from **HDF5Model** class:

```
public int H5Fcreate_wrap(String filename, int access_mode,
   int create_id, int access_id, boolean verbose)
```

This is helper method creating new HDF5 file.
Parameters:
filename File name
access_mode File create mode
create_id Create identifier
access_id Access identifier
verbose Verbose mode toggle
Returns:
File identifier

Example of use:

```
HDF5Model proc = new HDF5Model();
int file_id = proc.H5Fcreate_wrap(outputFileName,
   HDF5Constants.H5F_ACC_TRUNC, HDF5Constants.H5P_DEFAULT,
   HDF5Constants.H5P_DEFAULT, true);
```

3.2 Creating new HDF5 group

To create new group use **H5Gcreate_wrap** method from **HDF5Model** class:

```
public int H5Gcreate_wrap(int file_id, String name, int size_hint,
    boolean verbose)
```

This is helper method creating new HDF5 group.
Parameters:
file_id File identifier
name Group name
size_hint Size hint
verbose Verbose mode toggle
Returns:
Group identifier

Example of use:

```
HDF5Model proc = new HDF5Model();
int child_group_id = proc.H5Gcreate_wrap(file_id, "/what", 0,
    true);
```

3.3 Creating new HDF5 attribute

To create new attribute use **H5Acreate_any_wrap** method from **HDF5Model** class:

```
public void H5Acreate_any_wrap(int group_id, String attr_name,
   String attr_class, String attr_value, boolean verbose)
```

This is helper method for creating and writing HDF5 attribute of given type.
Parameters:

group_id Group identifier
attr_name HDF5 attribute name
attr_class HDF5 attribute data type
attr_value Access identifier
verbose Verbose mode toggle

Example of use:

```
HDF5Model proc = new HDF5Model();
proc.H5Acreate_any_wrap(child_group_id, "object", "string",
   "PVOL", true);
proc.H5Acreate_any_wrap(child_group_id, "nrays", "long",
    "360", true);
```

3.4 Creating new HDF5 simple dataspace

To create new simple dataspace use **H5Screate_simple_wrap** method from **HDF5Model** class:

```
public int H5Screate_simple_wrap(int rank, int dim_x, int dim_y,
    long maxdims[], boolean verbose)
```

This is helper method creating new HDF5 simple dataspace.
Parameters:

rank Rank
dims Dataspace dimensions
maxdims Maximum dataspace dimensions
dim_x Dataspace x dimension
dim_y Dataspace y dimension
verbose Verbose mode toggle

Returns:
Dataspace identifier

Example of use:

```
HDF5Model proc = new HDF5Model();
int dataspace_id = proc.H5Screate_simple_wrap(2, 360, 250, null,
   true);
```

3.5 Creating new HDF5 dataset

To create new dataset use **H5Dcreate_wrap** method from **HDF5Model** class:

```
public int H5Dcreate_wrap(int file_id, String group_name,
   int datatype_id, int dataspace_id, long[] chunk, int gZipLevel,
   boolean verbose)
```

This is helper method creating new HDF5 simple dataset.
Parameters:
file_id File identifier
group_name Group name
datatype_id Datatype identifier
dataspace_id Dataspace identifier
chunk A 2-D array containing the size of each chunk
gZipLevel Gzip compression level
verbose Verbose mode toggle
Returns:
Dataset identifier

Example of use:

```
HDF5Model proc = new HDF5Model();
longchunk[] = new long[2];
chunk[0] = rays;
chunk[1] = bins;
int grandgrandchild_group_id = proc.H5Dcreate_wrap(grandchild_group_id,
"data", HDF5Constants.H5T_STD_U8BE, dataspace_id, chunk, 6, verbose);
```

3.6 Writing HDF5 dataset

To write HDF5 dataset use following method:

```
public int H5Dwrite_wrap(int dataset_id, int mem_type_id,
   int mem_space_id, int file_space_id, int xfer_plist_id, Object buf,
   boolean verbose)
```

Helper method for writing HDF5 dataset.
Parameters:
dataset_id Dataset identifier
mem_type_id Memory type identifier
mem_space_id Memory space identifier
file_space_id File space identifier
xfer_plist_id
buf Data buffer
verbose Verbose mode toggle
Returns:
Operation status

Example of use:

```
HDF5Model proc = new HDF5Model();
proc.H5Dwrite_wrap(grandgrandchild_group_id,
   HDF5Constants.H5T_NATIVE_INT, HDF5Constants.H5S_ALL,
   HDF5Constants.H5S_ALL, HDF5Constants.H5P_DEFAULT,
   infDataBuff, verbose);
```

4 Troubleshooting

To report a bug please send information to lukasz.wojtas@imgw.pl

5 Major Improvements and bug fixes

Version 2.6 (Release date: 2010-06-10)

- Added Baltrad feeder.
- Added direct HDF5 converter.

Version 2.11 (Release date: 2010-11-15)

- Sending files by FTP.
- RAINBOW 5.31.1 format supported.

Version 2.12 (Release date: 2010-12-29)

- HDF5 to Rainbow format conversion.

Version 2.14 (Release date: 2011-02-09)

- Reading Quality Index from Rainbow volume files supported.

Version 2.14a (Release date: 2011-04-26)

- New manual.
- Missing “/Conventions” tag added.
- File compression optimized.