A new API for accessing ODV data collections from C++ and Java

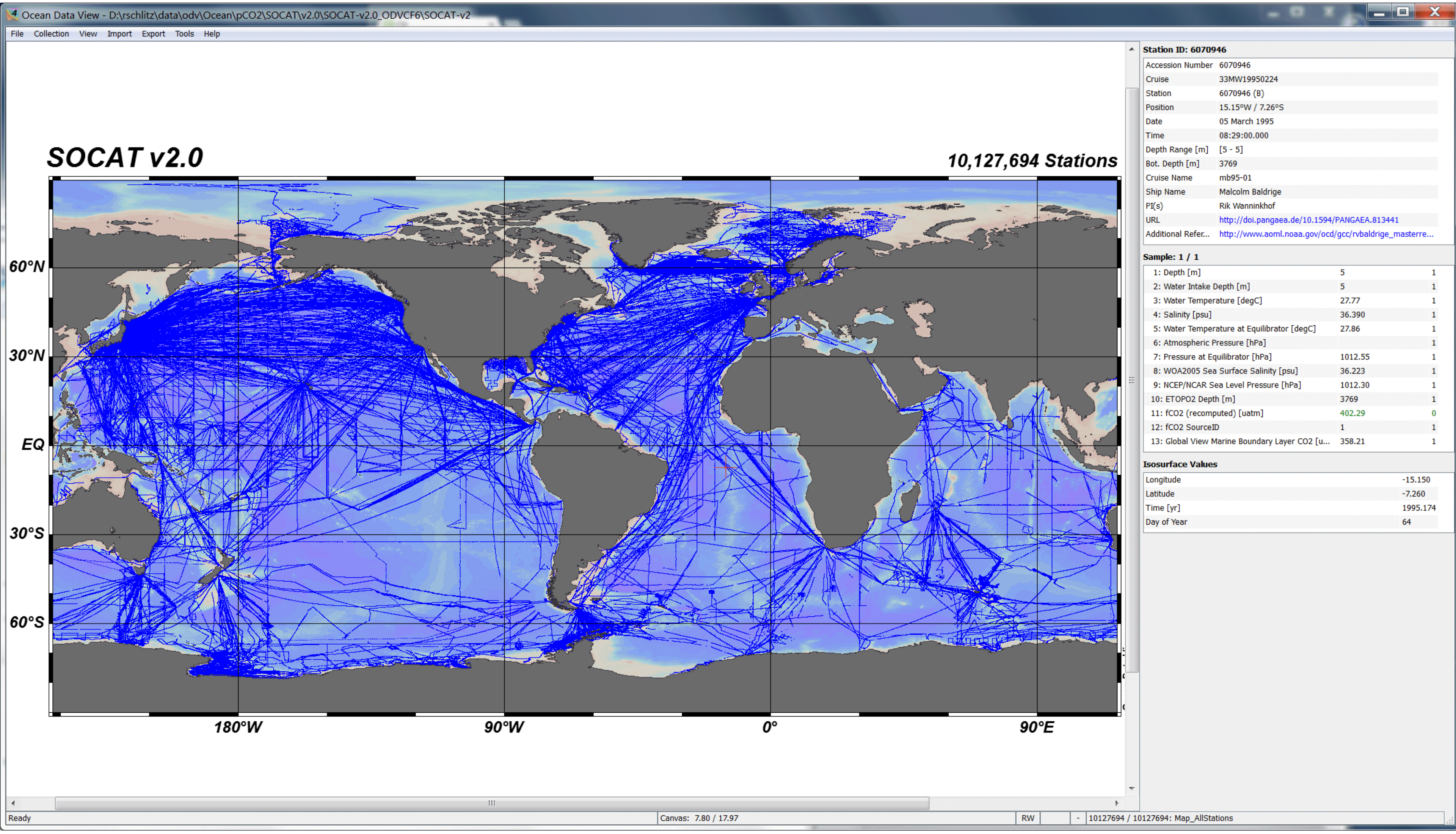
Reiner Schlitzer, Michael Menzel and Stephan Heckendorff

Alfred Wegener Institute for Polar and Marine Research

ODV collection format…

- ***Accomodates many data types***
- ***Provides dense storage***
- ***Allows very fast random access***
- ***Is platform independent***

Large ODV collections: SOCAT v2 pCO_2 data


Ocean Data View - D:\rschlitz\data\odv\Ocean\pCO2\SOCAT\v2.0\SOCAT-v2.0_ODVCF6\SOCAT-v2
File Collection View Import Export Tools Help
SOCAT v2.0
10,127,694 Stations
60°N
30°N
EQ
30°S
60°S
180°W
90°W
0°
90°E
Ready
Canvas: 7.80 / 17.97
RW
10127694 / 10127694: Map_AllStations

Station ID: 6070946

Accession Number	6070946
Cruise	33MW19950224
Station	6070946 (B)
Position	15.15°W / 7.26°S
Date	05 March 1995
Time	08:29:00.000
Depth Range [m]	[5 - 5]
Bot. Depth [m]	3769
Cruise Name	mb95-01
Ship Name	Malcolm Baldrige
PI(s)	Rik Wanninkhof
URL	http://doi.pangaea.de/10.1594/PANGAEA.813441
Additional Refer...	http://www.aoml.noaa.gov/ocd/gcc/rvbaldrige_masterre...

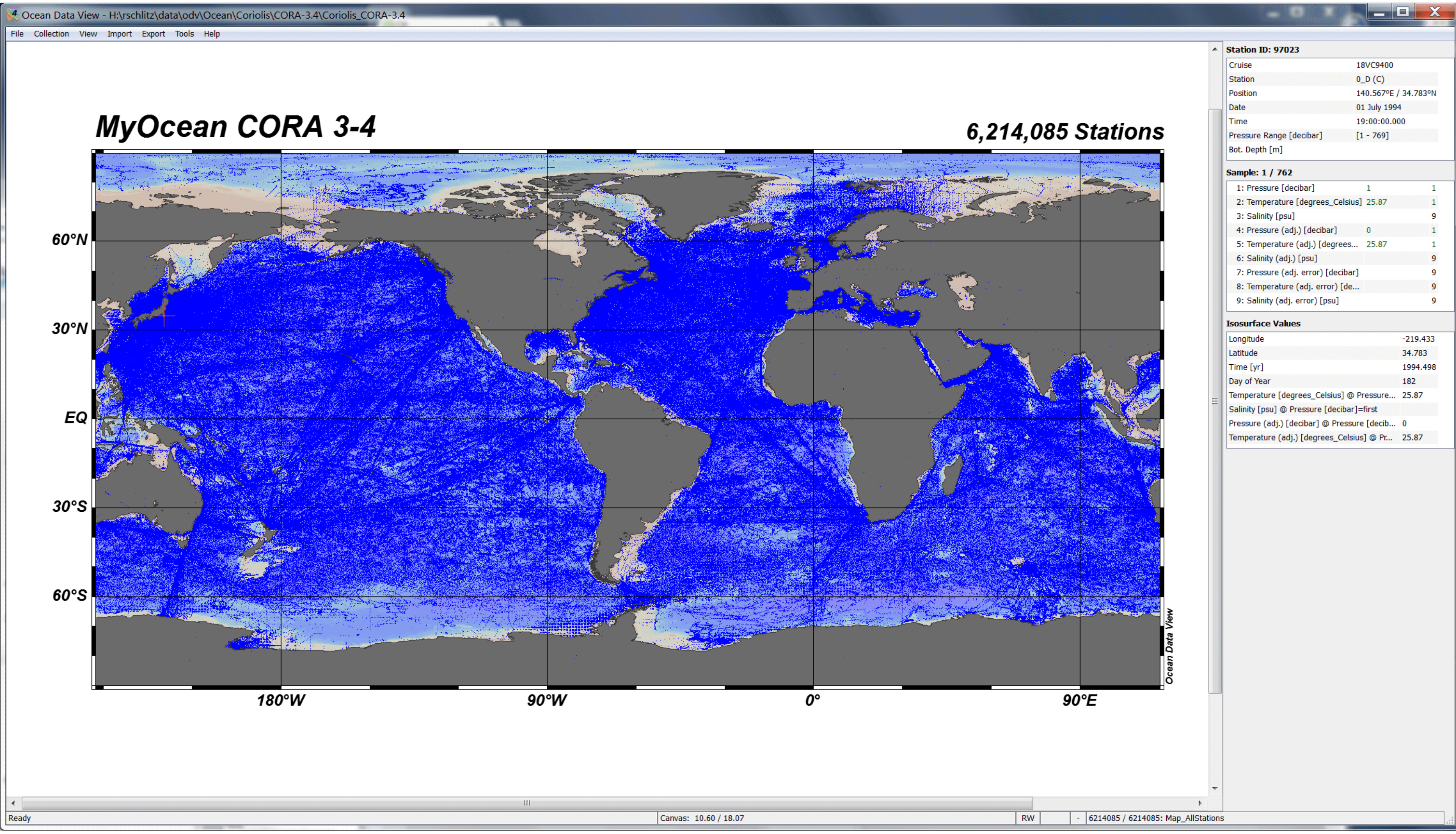
Sample: 1 / 1

1: Depth [m]	5	1
2: Water Intake Depth [m]	5	1
3: Water Temperature [degC]	27.77	1
4: Salinity [psu]	36.390	1
5: Water Temperature at Equilibrator [degC]	27.86	1
6: Atmospheric Pressure [hPa]		1
7: Pressure at Equilibrator [hPa]	1012.55	1
8: WOA2005 Sea Surface Salinity [psu]	36.223	1
9: NCEP/NCAR Sea Level Pressure [hPa]	1012.30	1
10: ETOPO2 Depth [m]	3769	1
11: fCO2 (recomputed) [uatm]	402.29	0
12: fCO2 SourceID	1	1
13: Global View Marine Boundary Layer CO2 [u...	358.21	1

Isosurface Values

Longitude	-15.150
Latitude	-7.260
Time [yr]	1995.174
Day of Year	64

Large ODV collections: MyOcean CORA3 T/S data


Ocean Data View - H:\rschlitz\data\odv\Ocean\Coriolis\CORA-3.4\Coriolis_CORA-3.4
File Collection View Import Export Tools Help
MyOcean CORA 3-4
6,214,085 Stations
60°N
30°N
EQ
30°S
60°S
180°W
90°W
0°
90°E
Ocean Data View
Ready
Canvas: 10.60 / 18.07
RW
6214085 / 6214085: Map_AllStations

Station ID: 97023

Cruise	18VC9400
Station	0_D (C)
Position	140.567°E / 34.783°N
Date	01 July 1994
Time	19:00:00.000
Pressure Range [decibar]	[1 - 769]
Bot. Depth [m]	

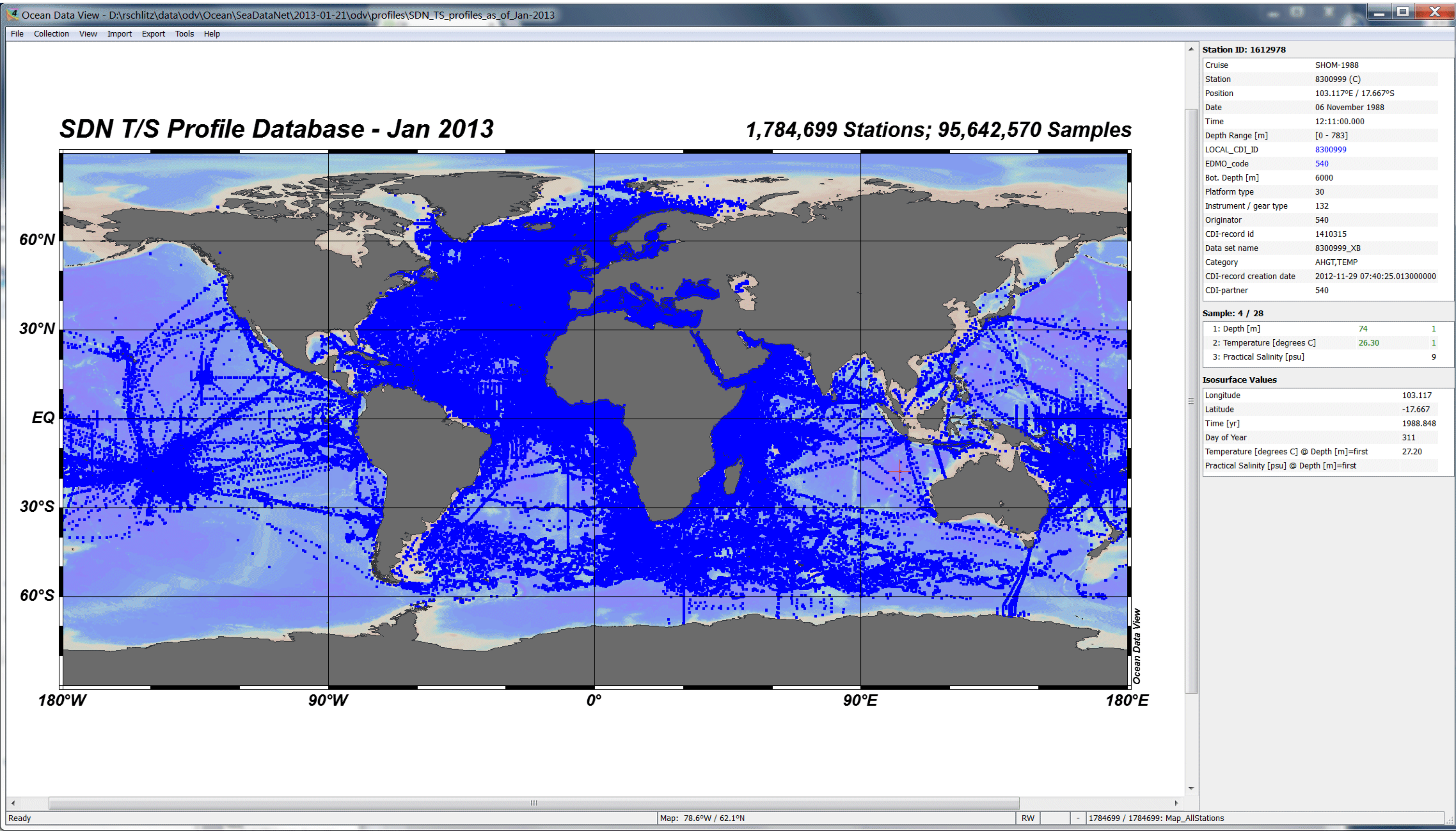
Sample: 1 / 762

1: Pressure [decibar]	1	1
2: Temperature [degrees_Celsius]	25.87	1
3: Salinity [psu]		9
4: Pressure (adj.) [decibar]	0	1
5: Temperature (adj.) [degrees...	25.87	1
6: Salinity (adj.) [psu]		9
7: Pressure (adj. error) [decibar]		9
8: Temperature (adj. error) [de...		9
9: Salinity (adj. error) [psu]		9

Isosurface Values

Longitude	-219.433
Latitude	34.783
Time [yr]	1994.498
Day of Year	182
Temperature [degrees_Celsius] @ Pressure...	25.87
Salinity [psu] @ Pressure [decibar]=first	
Pressure (adj.) [decibar] @ Pressure [decib...	0
Temperature (adj.) [degrees_Celsius] @ Pr...	25.87

Large ODV collections: SeaDataNet aggregated data


Ocean Data View - D:\rschlitz\data\odv\Ocean\SeaDataNet\2013-01-21\odv\profiles\SDN_TS_profiles_as_of_Jan-2013
File
Collection
View
Import
Export
Tools
Help
SDN T/S Profile Database - Jan 2013
1,784,699 Stations; 95,642,570 Samples
60°N
30°N
EQ
30°S
60°S
180°W
90°W
0°
90°E
180°E
Ocean Data View
Ready
Map: 78.6°W / 62.1°N
RW
1784699 / 1784699: Map_AllStations

Station ID: 1612978

Cruise	SHOM-1988
Station	8300999 (C)
Position	103.117°E / 17.667°S
Date	06 November 1988
Time	12:11:00.000
Depth Range [m]	[0 - 783]
LOCAL_CDI_ID	8300999
EDMO_code	540
Bot. Depth [m]	6000
Platform type	30
Instrument / gear type	132
Originator	540
CDI-record id	1410315
Data set name	8300999_XB
Category	AHGT,TEMP
CDI-record creation date	2012-11-29 07:40:25.013000000
CDI-partner	540

Sample: 4 / 28

1: Depth [m]	74	1
2: Temperature [degrees C]	26.30	1
3: Practical Salinity [psu]		9

Isosurface Values

Longitude	103.117
Latitude	-17.667
Time [yr]	1988.848
Day of Year	311
Temperature [degrees C] @ Depth [m]=first	27.20
Practical Salinity [psu] @ Depth [m]=first	

Past:

Data in ODV collections only accessible via Ocean Data View Software.

Future:

Provide ODV Application Programming Interface (API) allowing developers to create their own data access applications for …

- ***custom QC***
- ***scientific data processing***
- ***web-based data delivery***
- ***…***

Features provided:

- ***Read access to data, metadata and quality flag values in ODV collections (all formats, including the new ODVCF6)***
- ***ReadWrite access will be added in future version***
- ***User application code is platform independent (Windows, Mac OS X, Linux, Unix)***

What it consists of:

- *Set of header files providing the interface (ODV classes and functions; to be included in user code)*
- *Platform-specific compiled library to link against (Windows: odv4api.dll; Linux: libodv4api.so; Mac OSX: libodv4ap.dylib)*
- *Set of documentation and example files*
- *Supported languages: C++ (native), Java*

Key Concepts:

ODVCollection

- ***Holds values for arbitrary numbers of metadata and data variables for an arbitrary number of stations.***

ODVVariable

- ***Represents a metadata or data variable and holds its properties.***

ODVStation

- ***Represents sampling event with given space/time coordinates. Holds one value per metadata variable, and data values for every data variable and sample. Also holds quality flags for all metadata and data values.***

C++ Example Code (1/2)

```
/* create an ODVCollection object and open the collection in ReadOnly mode */
ODVCollection collection("c:/odv/test_collection.odv");
ODV::Status status=collection.open(ODV::ReadOnly);


/* retrieve number of metadata and data variables in collection */
int metaVarCount=collection.metaVarCount();
int dataVarCount=collection.basicVarCount();

/* obtain pointer to metadata variable varID (0-based index) */
ODVVariable *var=collection.metaVar(varID);

/* obtain pointer to data variable varID (0-based index) */
ODVVariable *var=collection.var(varID);

/* retrieve number of stations in collection */
int stationCount=collection.stationCount();


/* create an ODVStation object and read data of station statID in
   the collection. note that station IDs used in the readData() call
   are zero-based integers, e.g. 11 for 12th station. */
ODVStation station(&collection);
station.readData(statID);
```

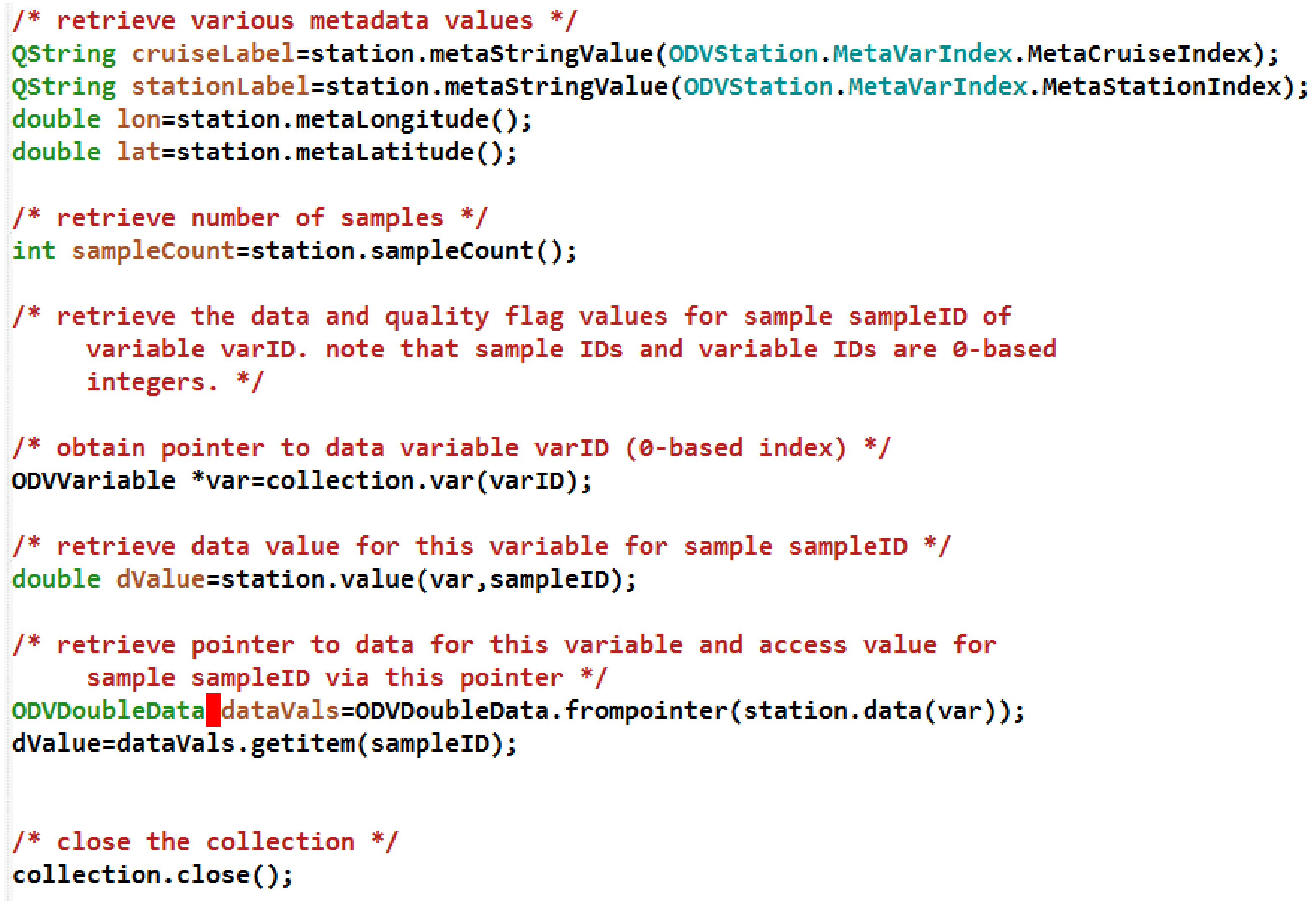
C++ Example Code (2/2)

```cpp
/* retrieve various metadata values */
QString cruiseLabel=station.metaStringValue(ODVStation::MetaCruiseIndex);
QString stationLabel=station.metaStringValue(ODVStation::MetaStationIndex);
double lon=station.metaLongitude();
double lat=station.metaLatitude();

/* retrieve number of samples */
int sampleCount=station.sampleCount();

/* retrieve the data and quality flag values for sample sampleID of
   variable varID. note that sample IDs and variable IDs are 0-based
   integers. */

/* obtain pointer to data variable varID (0-based index) */
ODVVariable *var=collection.var(varID);

/* retrieve data value and quality flag for this variable for sample sampleID */
char qFlag;
double dValue=station.value(var,sampleID,&qFlag);

/* retrieve pointer to data for this variable and access value for
   sample sampleID via this pointer */
double *dPtr=station.data(var);
dValue=dPtr[sampleID];


/* close the collection */
collection.close();
```

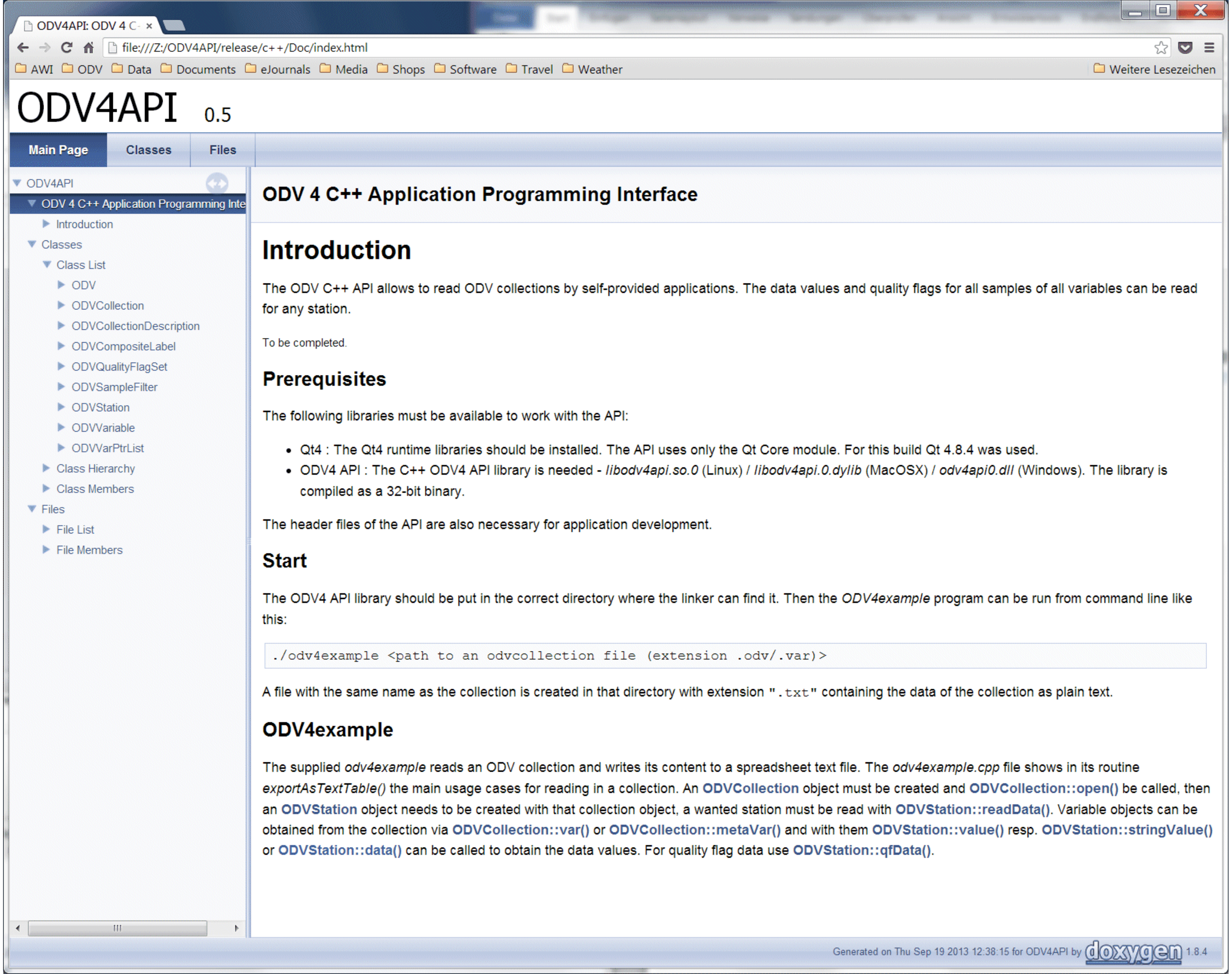
Java Example Code (1/2)

```java
/* import the odvapi java package */
import de.awi.odv.*;

...

/* load odv4 java api */
System.loadLibrary("odv4javaapi");

...

/* create an ODVCollection object and open the collection in ReadOnly mode */
ODVCollection collection=new ODVCollection(new QString("c:/odv/test_collection.odv"));
ODV.Status status=collection.open(ODV.AccessMode.ReadOnly);

/* retrieve number of metadata and data variables in collection */
int metaVarCount=collection.metaVarCount();
int dataVarCount=collection.basicVarCount();

/* obtain pointer to metadata variable varID (0-based index) */
ODVVariable *var=collection.metaVar(varID);

/* obtain pointer to data variable varID (0-based index) */
ODVVariable *var=collection.var(varID);

/* retrieve number of stations in collection */
int stationCount=collection.stationCount();


/* create an ODVStation object and read data of station statID in
     the collection. note that station IDs used in the readData() call
     are zero-based integers, e.g. 11 for 12th station. */
ODVStation station=new ODVStation(collection);
station.readData(statID);
```

Java Example Code (2/2)



```
/* retrieve various metadata values */
QString cruiseLabel=station.metaStringValue(ODVStation.MetaVarIndex.MetaCruiseIndex);
QString stationLabel=station.metaStringValue(ODVStation.MetaVarIndex.MetaStationIndex);
double lon=station.metaLongitude();
double lat=station.metaLatitude();

/* retrieve number of samples */
int sampleCount=station.sampleCount();

/* retrieve the data and quality flag values for sample sampleID of
     variable varID. note that sample IDs and variable IDs are 0-based
     integers. */

/* obtain pointer to data variable varID (0-based index) */
ODVVariable *var=collection.var(varID);

/* retrieve data value for this variable for sample sampleID */
double dValue=station.value(var,sampleID);

/* retrieve pointer to data for this variable and access value for
     sample sampleID via this pointer */
ODVDoubleData dataVals=ODVDoubleData.frompointer(station.data(var));
dValue=dataVals.getitem(sampleID);


/* close the collection */
collection.close();
```



ODV4API 0.5

ODV 4 C++ Application Programming Interface

Introduction

The ODV C++ API allows to read ODV collections by self-provided applications. The data values and quality flags for all samples of all variables can be read for any station.

To be completed.

Prerequisites

The following libraries must be available to work with the API:

- Qt4 : The Qt4 runtime libraries should be installed. The API uses only the Qt Core module. For this build Qt 4.8.4 was used.
- ODV4 API : The C++ ODV4 API library is needed - *libodv4api.so.0* (Linux) / *libodv4api.0.dylib* (MacOSX) / *odv4api0.dll* (Windows). The library is compiled as a 32-bit binary.

The header files of the API are also necessary for application development.

Start

The ODV4 API library should be put in the correct directory where the linker can find it. Then the *ODV4example* program can be run from command line like this:

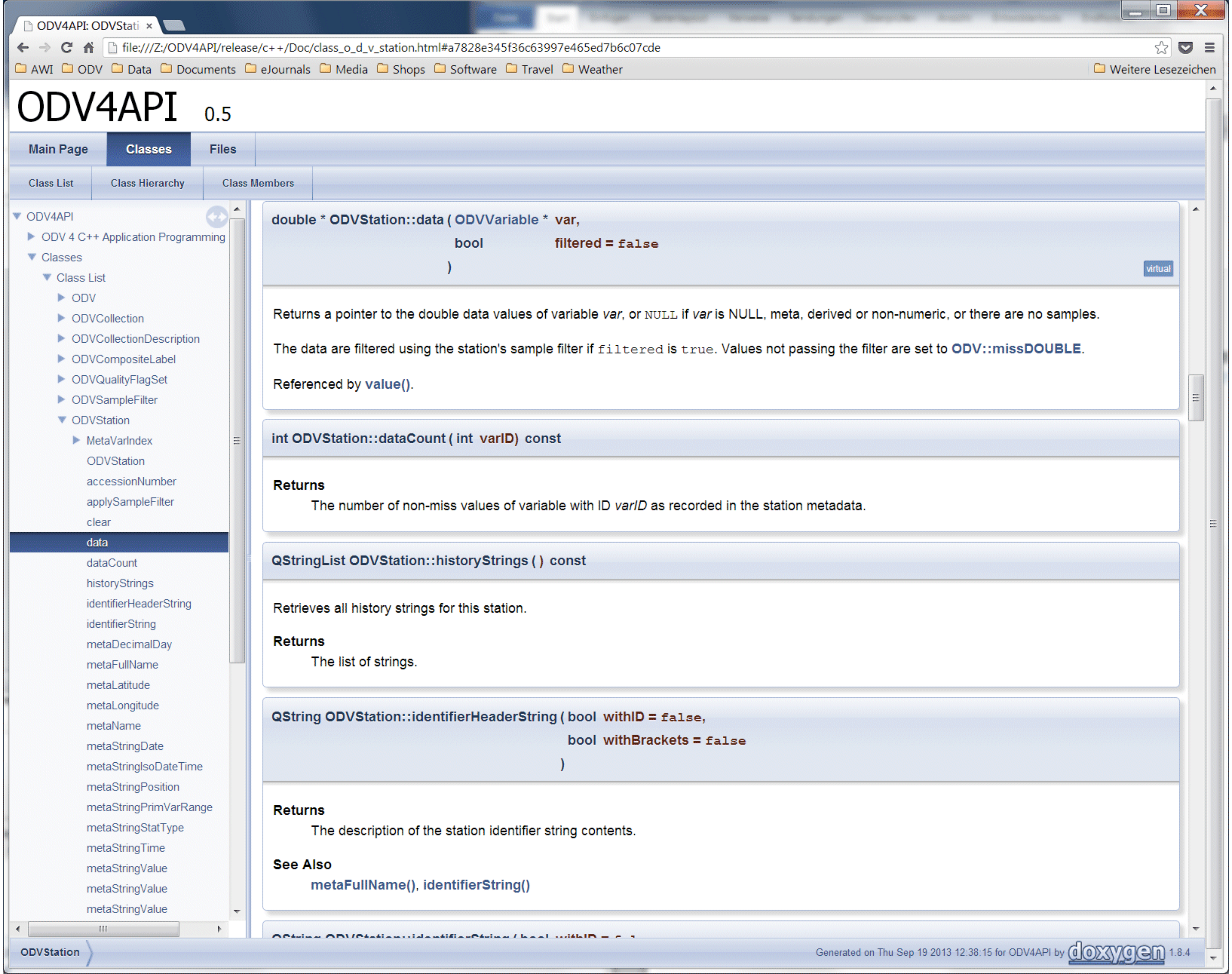
```
./odv4example <path to an odvcollection file (extension .odv/.var)>
```

A file with the same name as the collection is created in that directory with extension ".txt" containing the data of the collection as plain text.

ODV4example

The supplied *odv4example* reads an ODV collection and writes its content to a spreadsheet text file. The *odv4example.cpp* file shows in its routine *exportAsTextTable()* the main usage cases for reading in a collection. An **ODVCollection** object must be created and **ODVCollection::open()** be called, then an **ODVStation** object needs to be created with that collection object, a wanted station must be read with **ODVStation::readData()**. Variable objects can be obtained from the collection via **ODVCollection::var()** or **ODVCollection::metaVar()** and with them **ODVStation::value()** resp. **ODVStation::stringValue()** or **ODVStation::data()** can be called to obtain the data values. For quality flag data use **ODVStation::qfData()**.

Generated on Thu Sep 19 2013 12:38:15 for ODV4API by doxygen 1.8.4



ODV4API 0.5

```
double * ODVStation::data ( ODVVariable * var,
                            bool          filtered = false
                          )
```

virtual

Returns a pointer to the double data values of variable *var*, or `NULL` if *var* is NULL, meta, derived or non-numeric, or there are no samples.

The data are filtered using the station's sample filter if `filtered` is `true`. Values not passing the filter are set to **ODV::missDOUBLE**.

Referenced by value().

```
int ODVStation::dataCount ( int varID) const
```

Returns
The number of non-miss values of variable with ID *varID* as recorded in the station metadata.

```
QStringList ODVStation::historyStrings ( ) const
```

Retrieves all history strings for this station.

Returns
The list of strings.

```
QString ODVStation::identifierHeaderString ( bool withID = false,
                                             bool withBrackets = false
                                           )
```

Returns
The description of the station identifier string contents.

See Also
metaFullName(), identifierString()

ODVStation
Generated on Thu Sep 19 2013 12:38:15 for ODV4API by doxygen 1.8.4

License:

- *Similar to ODV license*
- *Usage free of charge for non-commercial research and teaching. Acknowledgement or citation required.*
- *Commercial usage requires license purchase.*

Availability:

- ***C++ and Java test versions available (personal contact or e-mail)***
- ***Beta status (tested in-house, Java version tested by IFREMER)***
- ***Planned release fall 2013, together with ODV 4.6.0***

Other Languages:

- ***Versions for Perl, PHP, Python, Tcl, Ruby, C#, R, Octave, GO or D can be produced using SWIG wrapper technology.***
- ***We seek cooperation with partners, if support for these additional languages is requested.***

Requires:

- ***Qt 4.8.4 must be installed (http://qt.digia.com/)***
- ***Can be used under LGPL license (e.g., free of charge if linking unmodified Qt dynamically)***
- ***Provides platform independence***