

---

## SOFTWARE METAPAPER

# BALTRAD Advanced Weather Radar Networking

Daniel Michelson<sup>1</sup>, Anders Henja<sup>2,3</sup>, Sander Ernes<sup>4</sup>, Günther Haase<sup>2</sup>, Jarmo Koistinen<sup>5</sup>, Katarzyna Ośródka<sup>6</sup>, Tuomas Peltonen<sup>7</sup>, Maciej Szewczykowski<sup>8</sup> and Jan Szturc<sup>6</sup>

<sup>1</sup> Environment and Climate Change Canada, Toronto, Ontario, CA

<sup>2</sup> Swedish Meteorological and Hydrological Institute, Norrköping, SE

<sup>3</sup> HENJAB, Växjö, SE

<sup>4</sup> IT and Development Centre, Ministry of the Interior, Tallinn, EE

<sup>5</sup> Finnish Meteorological Institute, Helsinki, FI

<sup>6</sup> Institute of Meteorology and Water Management, National Research Institute, Warsaw, PL

<sup>7</sup> Radiation and Nuclear Safety Authority, Helsinki, FI

<sup>8</sup> Brandpath Group, Warsaw, PL

Corresponding author: Daniel Michelson ([daniel.michelson@canada.ca](mailto:daniel.michelson@canada.ca))

---

BALTRAD software exchanges weather-radar data internationally, operationally, and in real-time, and it processes the data using a common toolbox of algorithms available to every node in the decentralized radar network. This approach enables each node to access and process its own and international data to meet its local needs. The software system is developed collaboratively by the BALTRAD partnership, mostly comprising the national Meteorological and Hydrological institutes in the European Union's Baltic Sea Region. The most important sub-systems are for data exchange, data management, scheduling and event handling, and data processing. C, Java, and Python languages are used depending on the sub-system, and sub-systems communicate using well-defined interfaces. Software is available from a dedicated Git server. BALTRAD software has been deployed throughout Europe and more recently in Canada.

---

**Keywords:** radar; weather radar; radar networking; data exchange; quality control

**Funding statement:** From 2009–2014, the BALTRAD and BALTRAD+ projects were part-financed by the European Union (European Regional Development Fund and European Neighbourhood and Partnership Instrument), with project numbers #009 and #101, respectively.

---

## (1) Overview

### Introduction

With the words “weather knows no boundaries”, eight organizations in seven countries sought and were granted funding in 2008 from the European Union's Baltic Sea Region (BSR) Programme, to network existing weather radars in the region. Weather radar is an active remote sensing technique that allows us to monitor the atmosphere with high resolution in both time and space. Such radars are deployed by the national meteorological (and hydrological) institutes of the respective countries. The radars operate at a frequency of around 5.6 GHz (5 cm wavelength), and have common ranges of 200–250 km, with new data being acquired every 5–15 minutes. Radar-based information helps save lives and property, and it facilitates operations in several weather-dependant sectors of the economy such as aviation, urban water management, radiation and nuclear safety, construction, and recreation.

The approved project, called BALTRAD (not an acronym), marked the first time European structure funds were used

to build an element of regional infrastructure in the form of an international weather radar network. BALTRAD also marked the first time that the project partners developed such a system together as a partnership. The BSR required that all outputs be made available openly and freely, which matched the partners' desire to release the software with an Open Source license. The lifetime of the BALTRAD project was three years (February 2009 to January 2012). Into the third year, the BSR approached the partnership and solicited a second proposal for an extension stage. The result was the BALTRAD+ project that saw both geographical and thematic expansion, now with 13 partners in all 10 BSR countries, and two additional years.

This paper provides a technical reference for the software that comprises a functional member of the network: a so-called “BALTRAD node”, along with an example user application. Following an introduction that provides the historical context, the system's main design constraints and system components are presented. Establishing a software framework will be highlighted

not just for real-time international data exchange but also for data processing. So will the importance of systematic data quality control and characterization, as this is the main scientific innovation achieved by the partnership.

Significant time has passed since the project funding ended, so the software's reuse potential will be discussed in the light of project vs. post-project realities.

### **Lead-up to BALTRAD**

Weather-radar data have been exchanged in real time in the Nordic collaboration called NORDRAD since the early 1990's [1, 2]. The first version of the system was developed using the Ada and Pascal programming languages, and it was deployed on VAX-VMS platforms. Data were exchanged over the X.25 protocol using a "notify-pull" mechanism whereby a node broadcasts data availability and those nodes entitled to such data are responsible for fetching the data. The software development and maintenance was performed by a commercial third party on behalf of the NORDRAD members. The first countries to deploy NORDRAD were Finland, Norway, and Sweden. Owing to low network bandwidth, the primary data that were exchanged were so-called CAPPI (Constant Altitude Plan Position Indicator) products, where the radar data's original spherical coordinate system expressed as elevation angle (from horizontal level), azimuth angle (from true north) and range (distance from the radar's location) is transformed to a regular Cartesian grid on a horizontal plane at a selected altitude. Each country reprocessed its own CAPPIs together with data from its neighbours, using the common software, to create so-called composite products covering large international domains using a projection and coverage area that was adapted to that country's needs. This is an example of a decentralized radar network concept.

In the early 2000's, the second generation NORDRAD system was commissioned, also by a commercial third-party. The decentralized networking concept was preserved, as was the basic data processing approach of exchanging and processing Cartesian data. There were two innovations with this new system: 1) it was Python-based for deployment on Linux platforms with communications using the TCP/IP protocol, and 2) algorithmic improvements from research experiences in the Baltic Sea Experiment [3] were pulled through to operations [4, 5]. Yet it soon became evident that improved network bandwidth was sufficient to support the exchange of original radar data in spherical coordinates, and that there was much greater potential in harmonizing data quality by accessing and processing such data. At the same time it became increasingly clear to the members, now including Denmark, Estonia, and Latvia, that such functionality could be created collaboratively. An important driving force in the weather radar world, for open and harmonized post-measurement data processing, is the fact that the quality challenges of the measured data are relatively large, especially as great variety exists among the technical and signal-processing

properties of the radar systems that establish international networks. Moreover, many weather radar operators and service providers are not able to put major resources in the R&D of new methodologies and algorithms. This led to the BALTRAD and BALTRAD+ projects that acted as development incubators, with a critical mass of resources devoted to establishing the partnership, software system, and weather radar network.

As has been noted in [6], BALTRAD is one of several open source radar software systems that have emerged in recent years. TITAN [7] is a system that pioneered the concept of open software for weather radar, and like BALTRAD is deployed in real-time operational environments. Py-ART [8] and wradlib [9] are two other packages that have a critical mass of developers and users, mainly for research and development purposes, but also for processing large amounts of archived data.

### **Implementation and architecture**

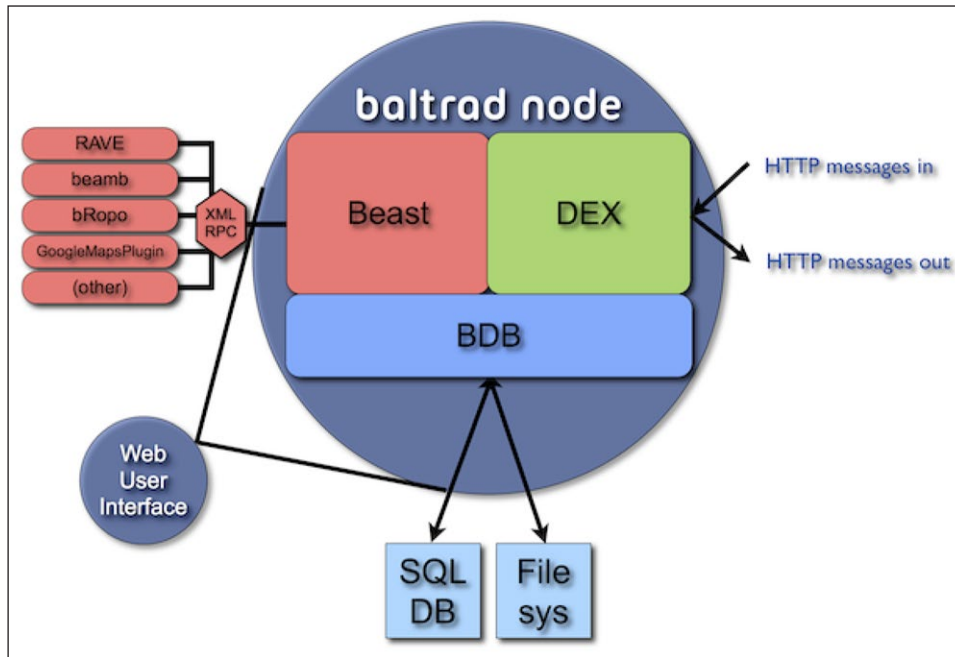
BALTRAD software is designed to exchange weather radar data in real time. Optionally, a sub-system is responsible for processing data using a common toolbox of algorithms. The same de-centralized networking concept as was applied in NORDRAD has also been used with BALTRAD, such that no individual partner is dependent on another to process data and disseminate radar-based products. A BALTRAD **node** is an instance of a complete system containing the functionality to exchange and, if desired, process data. BALTRAD software does not interact directly with the weather radar instrument. When the Partnership was formed, and its skill set inventoried, it was found that around half of the developers preferred Python and the other half preferred Java, while there were no objections to C/C++. So the decision was made to create functional sub-systems in any of these languages and ensure that they communicated with each other through well-defined interfaces.

A small package called the **node-installer** installs a complete functional BALTRAD **node**. It downloads specific versions of its dependencies (listed below), and provides a robust framework for installing and performing a basic configuration of a **node**. What follows are brief descriptions of each of the main sub-systems comprising a **node**, and how they interact. These sub-systems are illustrated in **Figure 1**.

### **Data exchange (DEX)**

The DEX contains the functionality used to subscribe to and exchange data in real time. In contrast to the NORDRAD "notify-pull" exchange mechanism, we use a "subscribe-push" model over secure network communications. Recalling that the partnership comprises different legal entities in different sovereign states, it is important that the communications contain sufficient security and checks and balances. To summarize:

1. Two **nodes** exchange secure keys, allowing them to transact securely.
2. A sender **node** selectively exposes its data catalogue to another receiver **node**.



**Figure 1:** BALTRAD node subsystems. The BALTRAD Toolbox is labeled RAVE and is shown on the left together with some of its add-on packages.

3. The receiver **node** selects which data it wishes to subscribe to.
4. The sender **node** pushes subscribed data to the receiver **node** in real time.
5. The receiver **node** validates that it has received what it has requested.

This peer-to-peer functionality is achieved in Java using a Tomcat server container. All communications between **nodes** are conducted through a single open port.

All weather radar data are expected to be exchanged using the modern European HDF5-based standard representation [10], and it is each partner’s responsibility to convert its data to this ODIM\_H5 format before “injecting” the data into their **nodes**. The partnership has developed converters/injectors in Java and Python for these purposes.

**BALTRAD database (bdb)**

Each **node** operates a PostgreSQL database instance that is used to manage all data and metadata. A Python application is responsible for this. Exchanged data entering a **node** from the DEX are passed to the **bdb**, the metadata are extracted from ODIM\_H5 files to populate database tables, and the ODIM\_H5 files themselves are preserved inside the database. This approach allows files to be associated with their metadata quickly, allowing for rapid access to input data by other parts of the **node**. ODIM\_H5 files can be optionally exported to local or remote file system. The **bdb** interacts with other sub-systems through a RESTful interface.

**Beast**

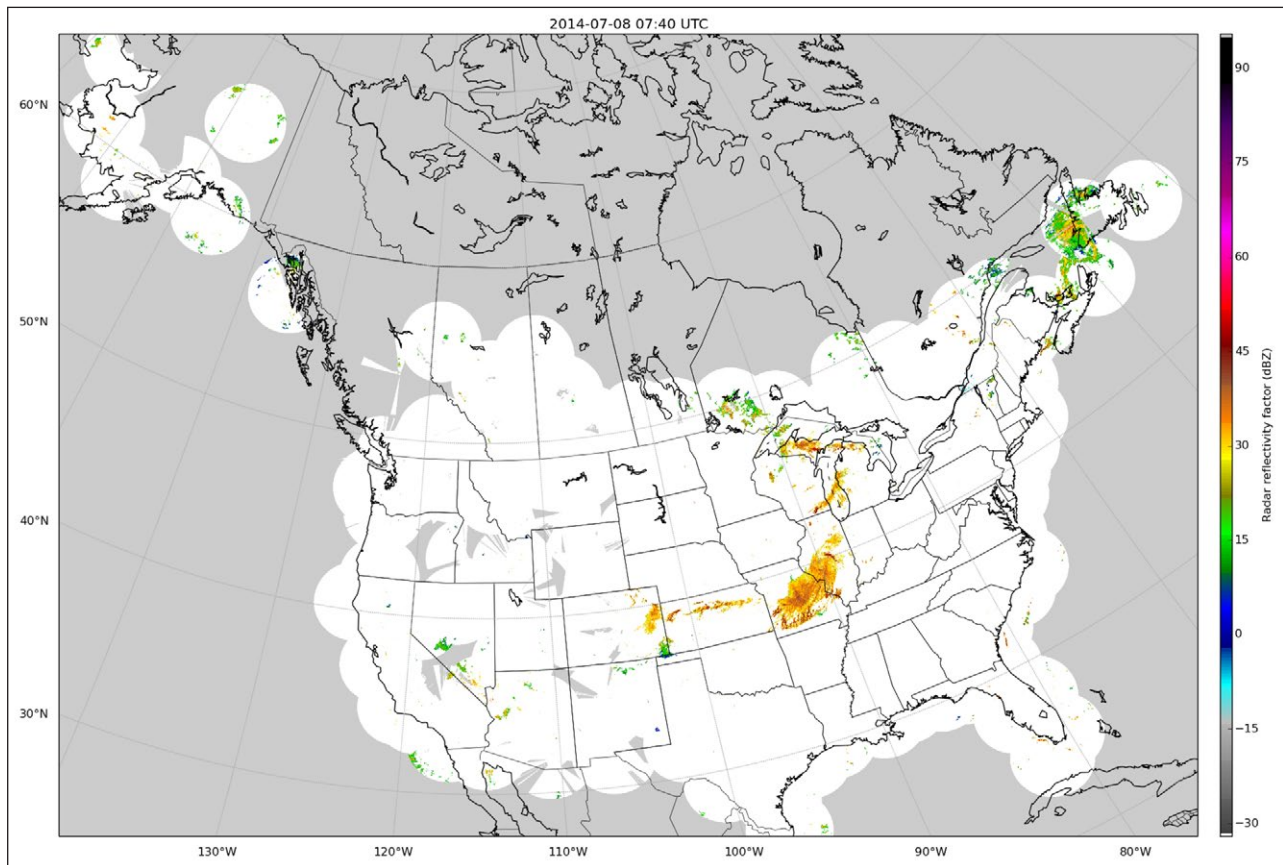
This sub-system is devoted to interacting with the other system components, to schedule event-based and

non-trivial data processing jobs, among other things, without actually running the jobs themselves. This Java application therefore interfaces with the **bdb** to learn when data are available to trigger such jobs, and they are triggered by messaging the **BALTRAD Toolbox**. A web-based interface is shared with the **DEX** for configuring and scheduling the **node** as a whole, along with different types of data processing jobs. Jobs are defined through this interface either using pre-defined rule-based menus, or through the use of **Groovy** scripting (based on Java).

An event-based job can be processing data from a single radar site, e.g. generating a Cartesian representation from the data’s original polar geometry. An example of non-trivial scheduling of a job is generating a composite image containing data from many radars; such a job is triggered either when all input data are available, or with those data that have arrived at a specified timeout.

**BALTRAD Toolbox**

The science of refining radar data into useful information is the responsibility of this software toolbox. It comprises a main package called the Radar Analysis and Visualization Environment (RAVE) along with several add-on packages for processing data for specific purposes. The code base is written in a combination of C and Python, to benefit from the high performance and relatively low memory footprint of C together with the ease of managing the system using Python. In real time, the **Toolbox** receives job requests from the **Beast** through an XML-RPC interface to its server, jobs are dispatched to an asynchronous pool of worker processes organized through Python’s **multiprocessing** module, and output is forwarded to the **DEX** and on to the **bdb**.



**Figure 2:** The BALTRAD Toolbox has been used with reflectivity data from 180 Canadian and American weather radars. Data have been quality controlled, and the derived quality information has been actively used when creating this composite product. July 8, 2014 at 07:40 UTC, showing storms over the American mid-west, the Great Lakes, and Atlantic Canada. Visualization has used Python's matplotlib and basemap packages on the Open Radar Virtual Machine [15].

Off-line work, e.g. algorithm development and processing archived data, is performed interactively through the Python interpreter, command-line utilities, or hands-off through scripting.

The **Toolbox** and its add-ons contains functionality for quality controlling radar data, e.g. identifying and removing non-precipitation echoes [11], correction due to topographical beam blockage, attenuation correction of conventional reflectivity data, data-quality characterization [12], and dealiasing of radial velocity data based on [13]. Compositing of data from many radars is supported in different ways, including the ability to use data quality as the main selection criterion (**Figure 2**).

The functionality for reading and writing data to ODIM\_H5 is separated from the data processing. This is done to enable the efficient chaining of processing algorithms in memory, without needing to write temporary/intermediate files to disk, which is achieved through a so-called "quality-plugin" framework.

As part of the extended thematic scope in BALTRAD+, we introduced the BALTRAD Cookbook (URL below) as a collection of openly documented data-processing algorithms, which can serve as a reference collection for implementation in our Toolbox or independently in other software.

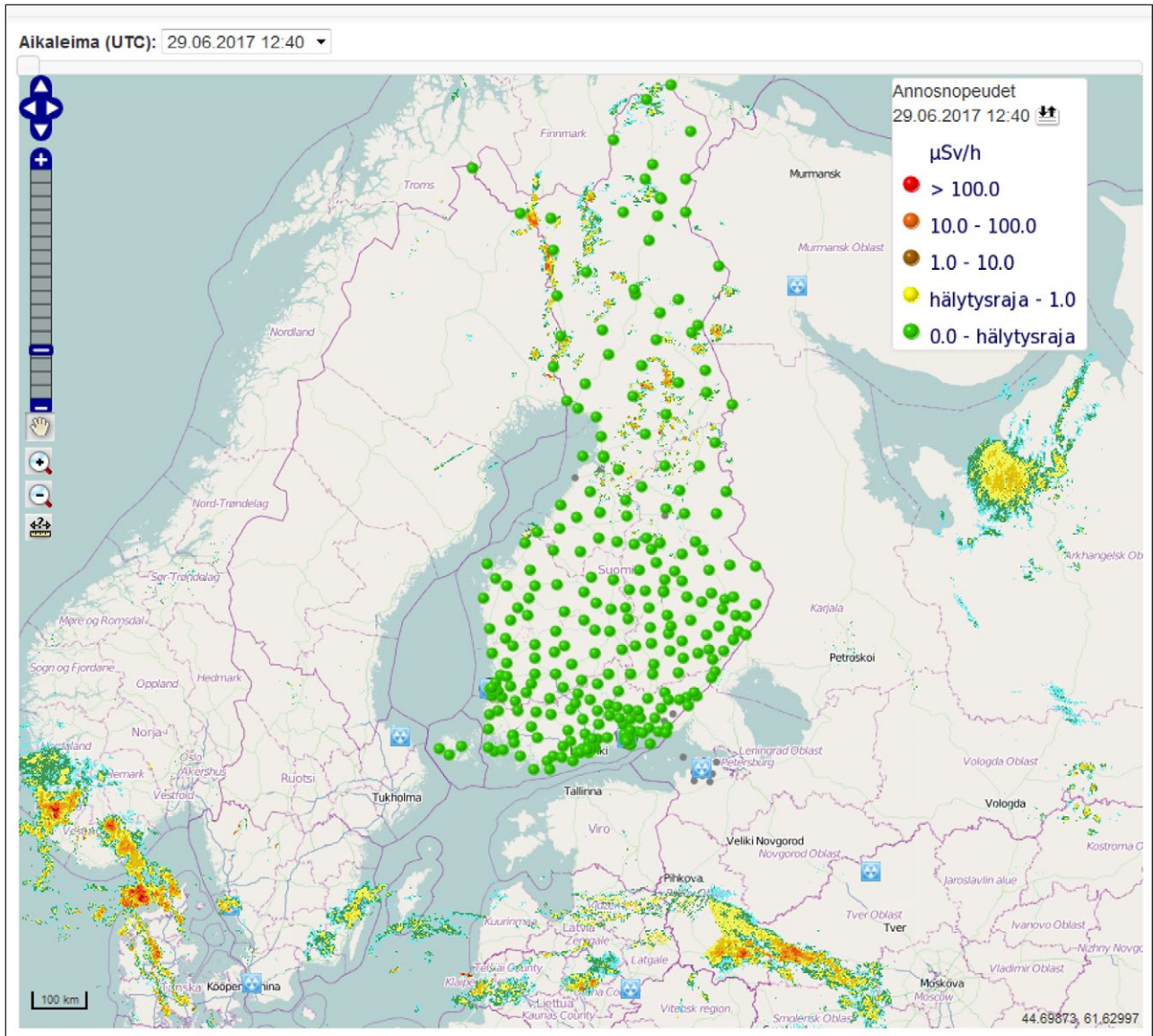
### Front-end applications

BALTRAD software is mostly intended to support various users, but we have also developed specific applications, one being a Web Map Services (WMS) server supporting radiation and nuclear safety in Finland, called **baltrad-wms** and based on an Open Source Geospatial Foundation (OSGeo) **MapServer**. Radar composites generated by a BALTRAD **node** are disseminated to a **baltrad-wms** instance where they are converted from ODIM\_H5 to GeoTIFF format and displayed (**Figure 3**).

### Quality control

Each of the sub-systems requires that unit tests be written. This is done using **JUnit** for Java and **unittest** for Python. Most developers apply agile methods when developing the system, to ensure that the system components achieve stable and consistent interfaces as early as possible, and to introduce new functionality in many small iterations. During the most intense development phase, lasting over a year, new versions of the **node-installer** were released and deployed live every two weeks. Project infrastructure supporting the development includes a dedicated **Git** server together with a **Jenkins** continuous integration server. Code coverage output is given for Java. Tickets are written and managed using a **Trac** server, which also hosts the BALTRAD Cookbook using its wiki functionality.





**Figure 3:** BALTRAD Web Map Services (baltrad-wms) instance displaying weather-radar reflectivity composite data together with the gamma dose-rate monitoring network operated in Finland for nuclear radiation safety.

Anyone is able to clone any of the software packages on the BALTRAD **Git** server; authorized users can push new commits. Each new commit triggers **Jenkins** to build the package and run its unit tests. The fundamental criterion that decides if new changes are acceptable is if all the unit tests pass; this will automatically merge the changes into the master branch. Jenkins also performs a nightly build and test run for the most important system components.

**(2) Availability**

**Operating system**

Linux, preferably 64-bit

**Programming languages**

- Gnu C/C++ 4.1+
- Python 2.6 and 2.7 (migration to 3.5 underway)
- Java 1.6+
- Javascript, Groovy

**Additional system requirements**

None

**Dependencies**

- PostgreSQL 8.4+
- Java SDK 1.6+
- Gnu make 3.8+
- Autoconf 2.59+
- Unicode support (ICU)
- bzip2 development libraries
- PostgreSQL development libraries
- Doxygen 1.4+

The following packages and their exact versions are managed by the BALTRAD **node-installer**, although more recent versions are also known to work.

- Apache ant 1.8.0
- Apache Tomcat 7.0.64

cURL 7.22.0  
 distribute 0.6.9  
 expat 2.0.1  
 HDF5 1.8.5-patch1  
 hdf-java 2.6.1  
 Python Imaging Library 1.1.7  
 NumPy 1.4.1  
 PIP 7.1.2  
 PROJ.4 4.7.0  
 ASN.1 Library for Python 0.1.2  
 PyCrypto 2.4.1  
 PycURL 7.19.5.3  
 Python 2.6.4 or 2.7.2  
 python-keyczar 0.7b  
 setuptools for Python 2.7  
 zlib 1.2.4

Additional optional dependencies are required for certain features.

#### List of contributors

- Rasphal Gill and Martin Sorensen, Danish Meteorological Institute, Copenhagen, Denmark
- Harri Hohti and Markus Peura, Finnish Meteorological Institute, Helsinki, Finland
- Iwan Holleman, Radboud University, Nijmegen, Netherlands
- Daniel Johnson, Ulf Nordh, Lars Norin, Mats Sundqvist, and Mats Vernersson, Swedish Meteorological and Hydrological Institute, Norrköping, Sweden
- Tomasz Sznajderski and Łukasz Wojtas, Institute of Meteorology and Water Management, Warsaw, Poland

#### Software location

##### Archive

(e.g. institutional repository, general repository)  
 (required – please see instructions on journal website for depositing archive copy of software in a suitable repository)

**Name:** Figshare

**Persistent identifier:** <https://doi.org/10.6084/m9.figshare.5901868.v1>

**Licence:** CC-BY 4.0

**Publisher:** Swedish Meteorological and Hydrological Institute, on behalf of the BALTRAD Partnership

**Version published:** 2.2.1

**Date published:** 19/02/18

##### Code repository

(e.g. SourceForge, GitHub etc.) (required)

**Name:** BALTRAD Git

**Identifier:** <http://git.baltrad.eu/git>

**Licence:** Lesser Gnu General Public Licence

**Date published:** 26/08/16

#### Language

English

### (3) Reuse potential

Externally-funded development projects are extremely useful in creating a critical mass of people and resources that can be used to create new outputs quickly and with

high priority; the BALTRAD and BALTRAD+ projects were successful in this regard. The critical moment comes when the projects end and the partner organizations assume responsibility for the operations, maintenance, and continued development of the project outputs. This is particularly sensitive if the outputs have been created by the partners themselves, and are not subject to any commercial service-level agreement. It is common for interest to wane at this point, so the challenge becomes one of ensuring that the resources are secured by each partner to continue contributing to the collaborative effort. The BALTRAD Cooperation Agreement has been written and signed by the partners to meet this challenge. The radar network continues to operate, and the software is being maintained and further developed. What follows are examples of not just potential reuse, but real reuse of BALTRAD outputs.

Before the end of the BALTRAD+ project, the Central Aerological Observatory of the Russian Federation joined the partnership, deployed a BALTRAD **node**, and started exchanging Russian weather-radar data with other partners. By the end of the project, data from 60 radars in ten countries were being exchanged, with data from eight additional specialized X-band radars in Denmark supporting urban hydrology.

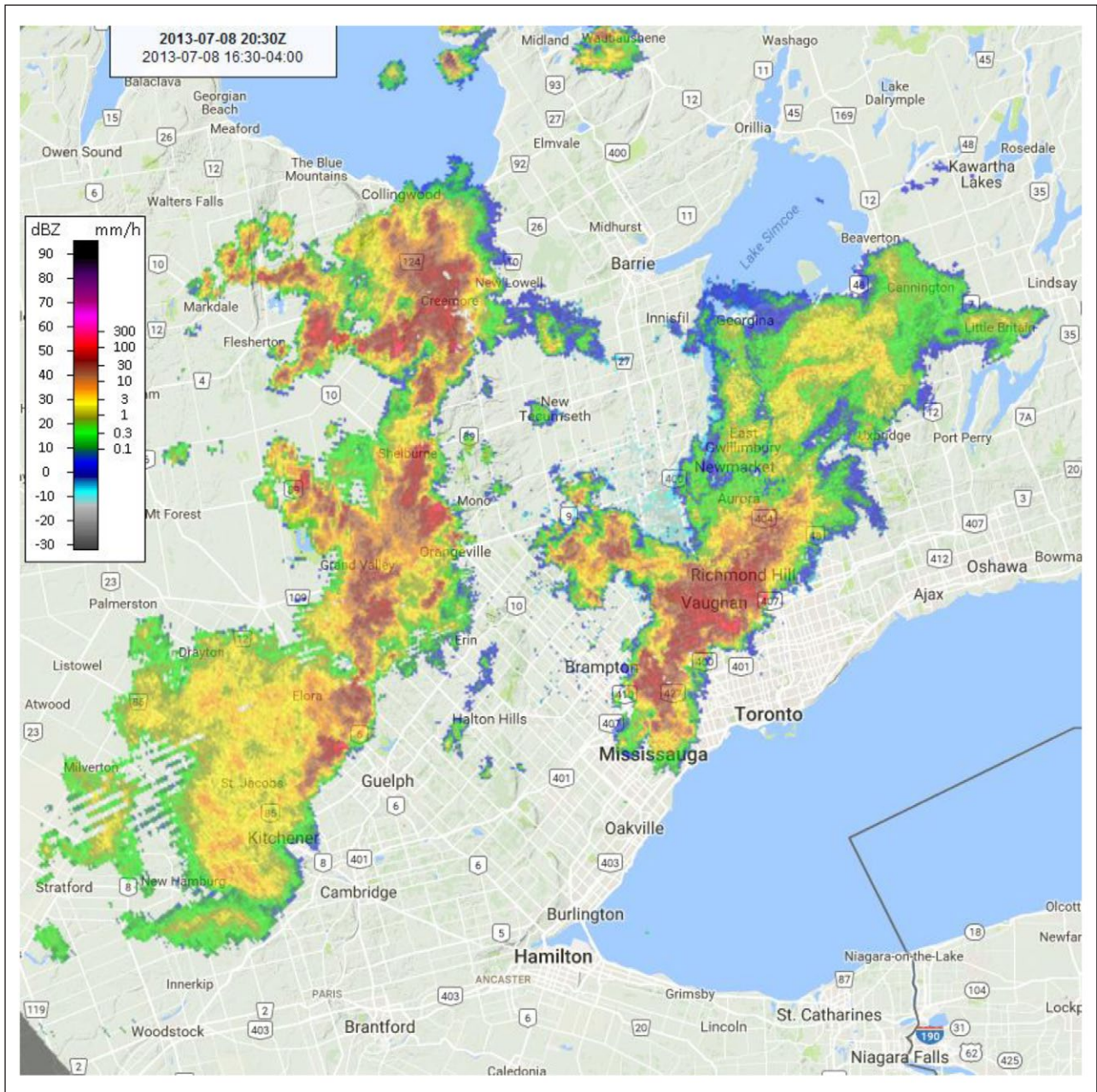
The network of European National Meteorological Services (EUMETNET) and its project for weather radar called OPERA [14] are responsible for a centralized European facility for collecting and generating European continental radar composite products. This service, called Odyssey and hosted jointly by Météo France and the United Kingdom Meteorological Office, has deployed the BALTRAD Toolbox operationally to quality control all available input radar data before European composites are generated on behalf of its members.

In 2014, the first Open Source Radar Software short course was held in association with a major radar conference. Contributors created an Open Radar Virtual Machine in which software presented in [6] was installed and made openly and freely available [15]. This solution has been proven useful at this short course and others that have followed, through Jupyter notebooks that run BALTRAD, Py-ART [8] and wradlib [9] software and visualize the results. Interoperability between these packages is also highlighted at these courses, an example of which is given in **Figure 4**.

Recently, the BALTRAD **Toolbox** was deployed in Canada to support the assimilation of weather radar data in numerical weather prediction. File-format decodes were required for North American data to provide input data in ODIM\_H5, and the NEXRAD Level II decode from Py-ART [8] is another good example of reuse for this purpose. Once this was achieved, the Toolbox was able to process Canadian and American data with the advantages of the BALTRAD data-quality processing framework (**Figure 2**). This is, in principle, the same quality-control approach as that already taken in Europe [18].

Additional reuse potential is based on the free and open availability of the software, and through its availability as part of the Open Radar Virtual Machine, in operational





**Figure 4:** Reflectivity data from the polarimetric King City weather radar north of Toronto, Canada, have been corrected for attenuation at C band [16] through software interoperability between BALTRAD and Py-ART [8] as implemented on the Open Radar Virtual Machine [15]. Visualization uses the experimental interface from BALTRAD to Google Maps. This event on July 8, 2013, gave 138 mm of rain near Toronto Pearson International Airport [17] and flooding that caused an estimated CAD \$ 1 billion in property damage.

real-time environments, or in academic/educational settings.

Finally, the legacy NORDRAD system was decommissioned in early 2017, marking reliance on BALTRAD as the fully-operational system for weather-radar networking in the BSR.

**Support mechanisms**

- FAQ, User Guide, and rendered inline documentation: <http://git.baltrad.eu/>
- BALTRAD Cookbook: <http://git.baltrad.eu/trac/wiki/cookbook>
- Google Group: [baltrad@googlegroups.com](mailto:baltrad@googlegroups.com)
- Support: [support@baltrad.eu](mailto:support@baltrad.eu)

- BALTRAD and BALTRAD+ project information: <http://www.baltrad.eu/>
- Open Radar Virtual Machine: <http://openradarscience.org/vm-docs/>

**Acknowledgements**

The authors wish to acknowledge and thank members of the BALTRAD Partnership not listed above, who did not contribute to the code base but who have been instrumental in setting up and running BALTRAD nodes in their countries, testing the system, writing algorithm recipes for the BALTRAD Cookbook, developing applications based on BALTRAD output, and otherwise contributing to the overall effort.

### Competing Interests

The authors have no competing interests to declare.

### References


1. **Carlsson, I** 1995 NORDRAD – Weather Radar Network. See [2], 45–52.
2. **Collier, C G** (ed.) 1995 COST 75 Weather Radar Systems. Brussels: European Commission. EUR 1601 EN. 814.
3. **Raschke, E, Meywerk, J, Warrach, K, Andræ, U, Bergström, S, Beyrich, F, Bosveld, F, Bumke, K, Fortelius, C, Graham, L P, Gryning, S-E, Halldin, S, Hasse, L, Heikinheimo, M, Isemer, H-J, Jacob, D, Jauja, I, Karlsson, K-G, Keevallik, S, Koistinen, J, Lehmann, A, Liljebladh, B, Lobmeyr, M, Matthäus, W, Mengelkamp, T, Michelson, D B, Napiórkowski, J, Omstedt, A, Piechura, J, Rockel, B, Rubel, F, Ruprecht, E, Smedman, A-S and Stigebrandt, A** 2001 The Baltic Sea Experiment (BALTEX): A European Contribution to the Investigation of Energy and Water Cycle over a Large Drainage Basin. *Bull. Amer. Meteor. Soc.*, 82(11): 2389–2413. DOI: [https://doi.org/10.1175/1520-0477\(2001\)082<2389:TBSEBA>2.3.CO;2](https://doi.org/10.1175/1520-0477(2001)082<2389:TBSEBA>2.3.CO;2)
4. **Michelson, D B, Andersson, T, Koistinen, J, Collier, C G, Riedl, J, Szturc, J, Gjertsen, U, Nielsen, A and Overgaard, S** 2000 *BALTEX Radar Data Centre Products and their Methodologies*, 76. Reports Meteorology and Climatology RMK No. 90, SMHI, SE-601 76 Norrköping, Sweden.
5. **Koistinen, J and Michelson, D B** 2002 BALTEX Weather Radar-based Precipitation Products and their Accuracies. *Boreal Env. Res.*, 7(3): 253–263. ISSN 1239-6095.
6. **Heistermann, M, Collis, S, Dixon, M, Giangrande, S, Helmus, J, Kelley, B, Koistinen, J, Michelson, D, Peura, M, Pfaff, T and Wolff, D** 2014 The Emergence of Open-Source Software for the Weather Radar Community. *Bull. Am. Meteorol. Soc.*, 96(1): 117–128. DOI: <https://doi.org/10.1175/BAMS-D-13-00240.1>
7. **Dixon, M and Wiener, G** 1993 TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A radar-based methodology. *J. Atmos. Oceanic Technol.*, 10: 785–797. DOI: [https://doi.org/10.1175/1520-0426\(1993\)010<0785:TTITAA>2.0.CO;2](https://doi.org/10.1175/1520-0426(1993)010<0785:TTITAA>2.0.CO;2)
8. **Helmus, J J and Collis, S M** 2016 The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. *Journal of Open Research Software*, 4(1): e25. DOI: <https://doi.org/10.5334/jors.119>
9. **Heistermann, M, Jacobi, S and Pfaff, T** 2013 Technical Note: An open source library for processing weather radar data (wradlib). *Hydrol. Earth Syst. Sci.*, 17(2): 863–871. DOI: <https://doi.org/10.5194/hess-17-863-2013>
10. **Michelson, D, Lewandowski, R, Szweczykowski, M, Beekhuis, H and Haase, G** 2014 EUMETNET OPERA weather radar information model for implementation with the HDF5 file format, version 2.2. *EUMETNET OPERA Deliverable*, 38. [http://www.eumetnet.eu/sites/default/files/OPERA2014\\_O4\\_ODIM\\_H5-v2.2.pdf](http://www.eumetnet.eu/sites/default/files/OPERA2014_O4_ODIM_H5-v2.2.pdf).
11. **Peura, M** 2002 Computer vision methods for anomaly removal. *Proc. Second European Conf. on Radar Meteorology*, 312–317. Delft, Netherlands. Delft University of Technology. <http://copernicus.org/erad/online/erad-312.pdf>.
12. **Ośródko, K, Szturc, J and Jurczyk, A** 2014 Chain of data quality algorithms for 3-D single-polarization radar reflectivity (RADVOL-QC system). *Meteorol. Appl.*, 21: 256–270. DOI: <https://doi.org/10.1002/met.1323>
13. **Haase, G and Landelius, T** 2004 Dealiasing of Doppler Radar Velocities Using a Torus Mapping. *J. Atmos. Oceanic Technol.*, 21: 1566–1573. DOI: [https://doi.org/10.1175/1520-0426\(2004\)021<1566:DODRVU>2.0.CO;2](https://doi.org/10.1175/1520-0426(2004)021<1566:DODRVU>2.0.CO;2)
14. **Huuskonen, A, Saltikoff, E and Holleman, I** 2014 The Operational Weather Radar Network in Europe. *Bull. Am. Meteorol. Soc.*, 95(6). DOI: <https://doi.org/10.1175/BAMS-D-12-00216.1>
15. **Heistermann, M, Collis, S, Dixon, M, Helmus, J, Henja, A, Michelson, D and Pfaff, T** 2015 An Open Virtual Machine for Cross-Platform Weather Radar Science. *Bull. Am. Meteorol. Soc.*, 96: 1641–1645. DOI: <https://doi.org/10.1175/BAMS-D-14-00220.1>
16. **Gu, J-Y, Ryzhkov, A, Zhang, P, Neille, P, Knight, M, Wolf, B and Lee, D-I** 2011 Polarimetric Attenuation Correction in Heavy Rain at C Band. *J. Appl. Meteor. Climatol.*, 50: 39–58. DOI: <https://doi.org/10.1175/2010JAMC2258.1>
17. **Boodoo, S, Hudak, D, Ryzhkov, A, Zhang, P, Donaldson, N, Sills, D and Ried, J** 2015 Quantitative Precipitation Estimation from a C-Band Dual-Polarized Radar for the 8 July 2013 Flood in Toronto, Canada. *J. Hydrometeorol.*, 16: 2027–2044. DOI: <https://doi.org/10.1175/JHM-D-15-0003.1>
18. **Ridal, M and Dahlbom, M** 2017 Assimilation of Multinational Radar Reflectivity Data in a Mesoscale Model: A Proof of Concept. *J. Appl. Meteor. Climatol.*, 56: 1739–1751. DOI: <https://doi.org/10.1175/JAMC-D-16-0247.1>



**How to cite this article:** Michelson, D, Henja, A, Ernes, S, Haase, G, Koistinen, J, Ośródka, K, Peltonen, T, Szewczykowski, M and Szturc, J 2018 BALTRAD Advanced Weather Radar Networking. *Journal of Open Research Software* 6: 12, DOI: <https://doi.org/10.5334/jors.193>

**Submitted:** 12 September 2017 **Accepted:** 22 February 2018 **Published:** 19 March 2018

**Copyright:** © 2018 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See <http://creativecommons.org/licenses/by/4.0/>.

 *Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiq Press

OPEN ACCESS 