Designation: E1948 – 98 (Reapproved 2022)

Standard Guide for Analytical Data Interchange Protocol for Chromatographic Data[1]

This standard is issued under the fixed designation E1948; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

1. Scope

1.1 This guide covers the implementation of the Chromatographic Data Protocol in analytical software applications. Implementation of this protocol requires:

1.1.1 Specification E1947, which contains the full set of data definitions. The chromatographic data protocol is not based upon any specific implementation; it is designed to be independent of any particular implementation, so that implementations can change as technology evolves. The protocol is implemented in stages, to speed its acceptance through actual use.

1.1.2 Specification E1947 contains a full description of the contents of the data communications protocol, including the analytical information categories with data elements and their attributes for most aspects of chromatographic tests.

1.2 The Analytical Information Categories are a practical convenience for breaking down the standardization process into smaller, more manageable pieces. It is easier for developers to build consensus and produce working systems based on smaller information sets, without the burden and complexity of the hundreds of data elements contained in all the categories. The categories also assist vendors and end users in using the guide in their computing environments.

1.3 The NetCDF Data Interchange System is the container used to communicate data between applications in a way that is independent of both computer architectures and end-user applications. In essence, it is a special type of application designed for data interchange.

1.4 The Common Data Language (CDL) Template for Chromatography is a language specification of the chromatography dataset being interchanged. With the use of the NetCDF utilities, this human-readable template can be used to generate an equivalent binary file and the software subroutine calls needed for input and output of data in analytical applications.

1.5 *This international standard was developed in accordance with internationally recognized principles on standardization established in the Decision on Principles for the Development of International Standards, Guides and Recommendations issued by the World Trade Organization Technical Barriers to Trade (TBT) Committee.*

2. Referenced Documents

2.1 *ASTM Standard:*[2]

E1947 Specification for Analytical Data Interchange Protocol for Chromatographic Data

2.2 *Other Standard:*[3]

NetCDF User's Guide

3. Features of the Chromatographic Data Interchange Implementation Guide

3.1 The chromatographic data protocol consists of: (*1*) the data contents (what is being transferred), and (*2*) the data container (how it is being transferred).

3.2 *Analytical Information Categories*—To make the standardization process more manageable, five Analytical Information Categories were defined:

1. Raw Data
2. Final Results
3. Full Data Processing Method
4. Full Chemical Method
5. Good Laboratory Practices Information

3.2.1 This guide covers implementation of Raw Data (Category 1) and Final Results (Category 2). The development committee has completed implementation and testing of information in these categories.

3.3 *Chromatography Data Elements*—Each data element specifies a piece of information that is to be stored or transferred to another system. A data element has a name, a definition, and may have some default attributes, such as its

[1] This guide is under the jurisdiction of ASTM Committee E13 on Molecular Spectroscopy and Separation Science and is the direct responsibility of Subcommittee E13.15 on Analytical Data.

Current edition approved Nov. 1, 2022. Published November 2022. Originally approved in 1998. Last previous edition approved in 2014 as E1948 – 98 (2014). DOI: 10.1520/E1948-98R22.

[2] For referenced ASTM standards, visit the ASTM website, www.astm.org, or contact ASTM Customer Service at service@astm.org. For *Annual Book of ASTM Standards* volume information, refer to the standard's Document Summary page on the ASTM website.

[3] Available from Russell K. Rew, Unidata Program Center, University Corporation for Atomospheric Research, P. O. Box 3000, Boulder, CO 80307-3000, http://www2.ucar.edu.

datatype, the analytical information category to which it belongs, and whether it is mandatory or recommended. A data element may also reference other data elements.

3.4 *NetCDF Data Interchange System*—NetCDF is a data interchange system designed specifically for the requirements of scientific data. It was developed to meet the requirements of high-performance, large dataset interchange over networks, in a machine- and network-independent way. NetCDF is implemented in the portable C programming language, and is made to be available on all popular scientific computers and operating systems. It achieves platform-independence through the use of the XDR protocol for byte-stream encoding. XDR (eXternal Data Representation) was developed by Sun Microsystems as part of the Network File System (NFS) and is now in the public domain. NetCDF has a particular advantage over many other approaches, such as ASCII flat files or structured ASCII files, because it uses the abstraction of a data access interface or abstract Application Programming Interface (API). This API lets the application developer deal with data more in terms of logical entities, instead of having to think about specific details of encoding and decoding data. This guide specifies the use of version 2.0 or later, and the exchange software should determine only that the version of NetCDF is version 2.0 or greater.

3.5 *Common Data Language*—A NetCDF data file has both binary and human-readable representations. The human-readable form is an ASCII-encoded file written in the NetCDF system's Common Data Language (CDL). CDL is a data description language, which is a simplified type of programming language without any action statements. The main purpose of a data description language is the specification of data structures. CDL is a generic data description language for scientific data; it is not application-specific. The basic construct in CDL is that of a multidimensional array, to which many types of metadata can be attached. Metadata is additional data about the data, that is, attributes about the data that help turn it into useful information.

3.5.1 For the protocol, a template for chromatography written in CDL is supplied to developers. The CDL template provides an easily comprehended text version of the structure and contents of a binary NetCDF file. Developers use this template with certain NetCDF utilities to generate the software code (subroutine calls) needed to use the NetCDF toolkit. The generated code can then be cut and pasted in the end-user application.

3.5.2 For a full description of the Common Data Language, see the NetCDF User's Guide.

3.6 *NetCDF Utilities*—The NetCDF data interchange system has two very powerful utilities called "ncgen" and "ncdump" that eliminate much work for developers.

3.6.1 After the template for a particular analytical technique dataset (in this case chromatography) is written in CDL, it is fed into ncgen. There are several options with ncgen, the first of which creates the binary file and populates it with fields specified by the CDL template. Other ncgen options generate code for the C language subroutines required for an application to read and write data in the NetCDF file, and the FORTRAN wrapper code needed for FORTRAN programs. The ncdump utility reads the binary files generated by ncgen (which were then populated by the application using the NetCDF libraries), and then generates their ASCII-encoded, human-readable representation. ncdump is routinely used to check the contents of NetCDF files, and debug applications that use NetCDF. It can also be used to export data to reporting programs. Fig. 1 illustrates the usage of the NetCDF utilities.

3.6.2 Developers typically start with the CDL source code description of the data file they wish to interchange. Then the ncgen utility with the -n option generates the NetCDF binary file. Developers then use ncgen with the -c option to generate the C source code needed for the subroutines to read and write this particular dataset (specified by the CDL template.) These read/write subroutines make up the NetCDF Application Programming Interface (API). These routines are simply cut and pasted into the target application(s). The programmer then uses the API to read or write data in NetCDF datasets.

3.6.3 If FORTRAN is the programming language being used, the ncgen utility with the -f option can generate the FORTRAN source code wrappers needed to embed the NetCDF library code into FORTRAN applications.

3.6.4 Once the application populates the data file with its data, that file can be transferred to another system. If a developer or end user needs a human-readable version of the binary dataset, the binary file can be fed into the ncdump utility, which regenerates the original CDL source code. The ncdump utility has an option switch to regenerate the CDL code with or without the actual experimental data. Some datasets can be enormous and there may be no reason to view the entire dataset. In this way, data are easily encoded into NetCDF files and the required routines to read and write those data are incorporated into applications.

3.6.5 There is another way to build NetCDF data files. Using the NetCDF Application Programming Interface, it is possible to build NetCDF binary files one data element at a time. However, that method is generally more time-consuming, and it is not recommended except for those situations where applications can fully automate the process.

4. Chromatographic Data Protocol Distribution Kit

4.1 Information on how to obtain the kit will be posted on the ASTM web site (www.astm.org) under Committee E49.

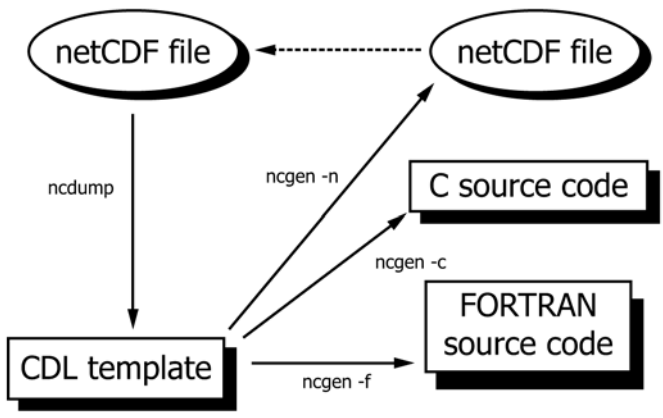

FIG. 1 Usage of NetCDF Utilities

4.2 The Analytical Data Interchange Protocol for Chromatographic Data Distribution Kit contains:

4.2.1 *Software Disks*—NetCDF distribution kit from Unidata (with the modified makefile needed to make the kit compile out of the box), and the Chromatographic Data Protocol CDL Template.

4.2.2 *NetCDF User's Guide*—supplied by Unidata Program Center.

4.2.3 *Specification E1947.*

4.2.4 *Guide E1948.*

4.2.5 *Registration Card,* used for tracking and user updates.

5. Hardware and Software

5.1 This section describes the hardware and software configurations used for testing. In general, the NetCDF system puts very few requirements on the hardware, because most routines are left on disk. Only routines being used at any particular time are kept in memory. Any limitations found were typically those not imposed by NetCDF but ones imposed by the operating system or environment.

5.1.1 *Hardware (Personal Computers)*—The personal computer system hardware used for testing was:

5.1.1.1 Intel 80286 processor,

5.1.1.2 640K minimum,

5.1.1.3 monochrome, EGA, VGA graphics,

5.1.1.4 20 megabyte minimum, 80 megabyte hard-disk is typical, and

5.1.1.5 a mouse (optional).

5.1.1.6 NetCDF works well on AT-class machines and higher. NetCDF does not have the items in 5.1.1.1 – 5.1.1.5 as requirements. These are just the minimum, base-level systems that were used.

5.1.2 *Software*—NetCDF runs on MS-DOS, OS/2, Macintosh, Windows 95, and Windows NT operating systems for personal computers. NetCDF was originally ported from UNIX to DOS running on an IBM-PS/2 Model 80. It was recently ported to the Macintosh OS. NetCDF is written is the C programming language, and there are FORTRAN jackets available for applications that want to use FORTRAN calls. The personal computer software employed for testing and developing NetCDF applications were:

5.1.2.1 Microsoft DOS V3.3 or above,

5.1.2.2 Microsoft C Compiler V6.0,

5.1.2.3 Microsoft Windows V3.0,

5.1.2.4 Microsoft Windows SDK, and

5.1.2.5 NetCDF Version 2.0.1.

5.1.3 *Workstations and Servers*—NetCDF runs easily on UNIX workstations such as Sun 3, Sun 4, VAXstations, DECstation 3100, VAXstation II running ULTRIX or VMS, and IBM RS/6000. There are no particular hardware requirements for workstation class machines, since all workstations have the minimum hardware outlined for personal computers in 5.1.1.

6. Significance and Use

6.1 *General Coding Guidelines*—The NetCDF libraries are supplied to developers as source code. End users receive the libraries in compiled binary form as part of a vendor's application.

6.1.1 Some vendors found that compilation using the huge memory model under Microsoft Windows on MS-DOS was needed because of an array pointer passing its boundary. Many vendors chose to write a conversion program as a stand-alone DOS application, whereby the conversion program used only text-mode screen I/O. Some vendors are now using it in their MS-Windows applications.

6.1.2 Developers setting out to write a program to convert their data files to the Chromatographic Data Protocol should use the NetCDF utilities ncgen and ncdump. Applying the ncgen utility to the CHROMSTD.CDL template will generate the skeletal code needed to create the NetCDF file. The programmer can then modify the code to create a program that reads the netDCF file from another vendor. After developers create the NetCDF file they should use the ncdump program to generate the ASCII representation of the data files, and examine it to ensure the data is being correctly put into the file.

6.2 *Make Files for NetCDF Libraries and Utilities*—In general the compilation is straightforward. The make files were modified after they were received from the Unidata Corporation, because they did not compile the first time on PCs. The changes needed to get the Unidata distribution to run on DOS are (*1*) rename the file MAKEFILE to UNIX.MK, and (*2*) rename MSOFT.MK to MAKEFILE, and then run NMAKE. The default switches in the Unidata distribution use the switches for the floating point coprocessor and Microsoft Windows options.

6.2.1 The protocol contain some complete makefile examples for Microsoft C V6.0 running on DOS. The Microsoft C V6.0 compiler manual should be consulted for the exact meaning of the compiler and linker options.

6.2.2 The VMS and SunOS compilation instructions are in directories for those operating systems.

6.3 *NetCDF Library Build Order*—The NetCDF libraries must be built in a specific order. The correct order to build the NetCDF directories is:

```
UTIL
XDR
SRC
NCDUMP
NCGEN
NCTEST
```

6.3.1 The UTIL and XDR makefiles work as distributed using NMAKE with Microsoft C V6.0.

7. CDL Template Structure

7.1 A NetCDF template is built from CDL statements and structured into three sections: (*1*) dimension declarations, (*2*) variable declarations, and (*3*) the data section.

7.2 A few points of clarification about the CDL language are given here to facilitate understanding of the CDL Generic Chromatography Template given in Fig. 2. For more in-depth information on CDL, please consult the NetCDF User's Guide.

7.2.1 A NetCDF template starts with the word "NetCDF" followed by the dataset name.

7.2.2 CDL comments are indicated by two forward slash characters (//).

```
netCDF AIA_chromatography_template

{

// File: chromstd12.cdl.      Date: 1-26-92
// CDL code for AIA Chrom netCDF file.  AIA Specification Version 1.0.
// AIA Specification Version 1.0 only includes Analytical Information
// Categories 1 & 2.
// It requires Version 2.0 of netCDF

// Copyright 1991 - The Analytical Instrument Association

dimensions:

// * The following dimensions predefine some useful character array
// * default string lengths.  The actual size of a character dimension is
// * arbitrary.  These were picked by considering computer string length
// * storage efficiency.  These dimension values should
// * not be changed by your data source program.

    _2_byte_string = 2;      // 2 character string length
    _4_byte_string = 4;      // 4 character string length
    _8_byte_string = 8;      // 8 character string length
    _16_byte_string = 16;    // 16 character string length
    _32_byte_string = 32;    // 32 character string length
    _64_byte_string = 64;    // 64 character string length
    _128_byte_string = 128;  // 128 character string length
    _255_byte_string = 255;  // 255 character string length

// * The following variables need to be set by your program.  The values below
// * are given just as examples.  The actual values will be
// * set by the data source program after it
// * does the peak integrations and computes how many peaks and components
// * there are.  This should be done before the netCDF file is written out
// * from your application.

    point_number = 7;        // point number dimension for raw data   M1
    peak_number = 1;         // number of peaks                        M2
    error_number = 1;

 variables:

// Administrative Information Class - Category 1,2 Data Elements
    :dataset_completeness            = "C1+C2";                 // M12345
// Categories 1&2, raw & result data
    :aia_template_revision           = "1.0";                   // M12345
    :netcdf_revision                 = "2.0";                   // M12345
```

FIG. 2 Template

```
    :languages                          = "English-only for now";
    :administrative_comments            = "none for now!";
    :dataset_origin                     = "AIA member company name"; // M5
    :dataset_owner                      = "AIA Companies";
    :dataset_date_time_stamp            = "19910901123030-0500";
    :injection_date_time_stamp          = "19910901123030-0500";     // M12345
    :experiment_title                   = "Working Demonstrations";
    :operator_name                      = "Joe Scibler";             // M5
    :separation_experiment_type         = "liquid chromatography";
    :company_method_name                = "sandope analysis XYZ";
    :company_method_id                  = "SAXYZ";
    :pre_experiment_program_name        = "setup";
    :post_experiment_program_name       = "response calibration";
    :source_file_reference              = "IODINE::dka100:[aia]test.cdl";
    char  error_log(error_number, _64_byte_string);

// SAMPLE-DESCRIPTION Information Class - Category 1 Data Elements

    :sample_id_comments                 = "none";
    :sample_id                          = "JOU812";
    :sample_name                        = "test sample";
    :sample_type                        = "control";
    :sample_injection_volume            = "2.0";
    :sample_amount                      = "2.0";

// DETECTION-METHOD Information Class - Category 1 Data Elements

    :detection_method_table_name        = "test 1";
    :detector_method_comments           = "An optical detector";
    :detection_method_name              = " ";
    :detector_name                      = "variable wavelength detector";

// the following information is used for proper raw data scaling

    float detector_maximum_value;                                   // M1
    float detector_minimum_value;                                   // M1
    :detector_unit                      = "Absorbance Unit";
// M1, e.g., A.U.

// RAW-DATA Information Class - Category 1 Data Elements

    :raw_data_table_name                          = "test raw data set";
    :retention_unit                               = "time in seconds";
                                                  // M12
    float actual_run_time_length;                 // M12, in retention_unit
    float actual_sampling_interval;               // M12, in retention_unit
    float actual_delay_time;                      // M12, in retention_unit

    float ordinate_values (point_number);         // M1
         ordinate_values:uniform_sampling_flag = "Y"; // M1
         ordinate_values:autosampler_position = "1.01";
                  // real_time autosampler position
    float raw_data_retention (point_number);
                  // M1 only if uniform_sampling_flag = "N"
```

FIG. 2 Template *(continued)*