RTP Format Specification and User's Guide

Version 2.21

Howard E. Motteler and Scott Hannon

June 12, 2024

Abstract

We present a data format for driving radiative transfer calculations and manipulating atmospheric profiles. Calculated and observed radiances may be included as optional fields, for more general applications. An implementation as HDF4 Vdatas is given, including Fortran, C, and Matlab application interfaces.

1 Introduction

The "Radiative Transfer Profile" (RTP) format is a data format for sets of atmospheric profiles, optionally paired with calculated and/or observed radiances. The format consists of a header record and an array of profile records. It was derived from the GENLN2 user profile format, extened with selected AIRS level 2 field definitions. RTP is implemented as HDF4 Vdatas.

The format is intended to give a well-defined interface to radiative transfer codes, allowing for the specification of just the information needed for such calculations. It allow for modularity of both radiative transfer codes and of other tools for manipulating profiles, including tools for field selection, level interpolation and level-to-layer translations, translation of units, and building composite profiles from multiple sources. The RTP specification has some flexibility in the field set actually saved to disk, both to save space and to provide compatibility across file versions. The optional observation fields may be used to build simple co-location datasets.

2 The RTP format definition

The RTP format consists of a header record with information about all the profiles in a file, and one or more profiles saved as an array of records. Field definitions for the header and profile records are given below. These names are both the names of the HDF4 Vdata fields and the Fortran, C, and Matlab structure fields, with the exception of the constituent arrays, as discussed below. Fields are matched by name, and depending on the application, only a subset of the fields described here need be present in an RTP file.

2.1 Levels and Layers

The header field `ptype` flags the profile as being a level profile, a layer profile, or a pseudo-level profile. For level profiles, the temperature and gas constituent fields represent point values at the specified pressure levels. For layer profiles these fields represent integrated values in the space between adjacent `plevs`. The `palts` field, if used, is altitudes of the pressure levels for either level or layer profiles. The `nlevs` field is the number of pressure levels. For layer profiles the number of layers is `nlevs` − 1. Pseudo-level profiles contain layer gas consistuents and level temperatures.

A convention that lower indices correspond to lower pressures is suggested but not required. The header fields `pmax` and `pmin` are intended to hold the max and min level pressures over all profiles, or some upper and lower bound on these values.

2.2 Constituents

Constituent fields are named with their HITRAN gas ID's, with `gas_1` water, `gas_2` CO_2, and so on. A list of HITRAN gas ID's is given in an appendix. The header field `glist` gives a list of the IDs for the constituents present in the file. The default constituent unit is PPMV.

The Fortran and C application interfaces represent constituents as a 2D array `gamnt` whose rows are layers and whose columns are gas ID index, rather than as a set of separate fields `gas_<i>` as they are actually saved in the file. The `gas_<i>` fields are the columns of the 2D `gamnt` array.

There are a wide variety of constituent units in current use; in consideration of this we have added a `gunit` array to the header, assigning a unit code for each constituent and allowing at least the potential for automatic conversions. These unit codes are given in `gas_units_code.txt`.

Note that only a small subset of possible constituents are typically recognized and processed by fast models for radiative transfer calculation, typically water, ozone, and perhaps methane, CO_2, and CO; see the documentation of the relevant radiative transfer code for more information.

2.3 Field Sets and Sizes

The RTP interface uses the header size fields `mlevs`, `memis`, `ngas`, `nchan`, and `pfields` to build new, generally much more compact, field lists for the header and profile Vdata buffers. Whenever a header size field is zero, the relevant header or profile field is dropped from the Vdata. The header size fields `mlevs`, `memis`, `ngas`, and `nchan` all default to zero, and should be less than or equal to the corresponding max limits `MAXLEV`, `MAXEMIS`, `MAXGAS`, and `MAXCHAN`.

Individual profiles have pressure levels set with `nevs`, and emissivity and reflectance sets with `nemis`, both of which default to zero. These should be less than or equal to the header limits `mlevs` and `memis`. For most applications they can just be set to `mlevs` and `memis`. All profiles in a file are assumed to have the same constituent set, and if radiances are present all profiles have the same channel set. Most arrays have an associated size field. If this size field is in the header, as in the case of `ngas` or `nchan` then it is assumed to be the same for all profiles, while if the size field is in a profile, as in the case of `nlevs` or `nemis`, then it applies only to that profile.

2.4 Field Groups

The `pfields` field in the header is used by the C/Fortran API to control what which field groups will be written to a file. Profile fields are organized as three groups,

```
1. profile data                    PROFBIT = 1
2. calculated radiances          IRCALCBIT = 2
3. observed radiances            IROBSVBIT = 4
```

These groups can occur in any combination. The associated numbers are bit fields, to be set in pfields if the associated data is present in the file. Thus for example profile data with calculated and observed radiances would be represented as `pfields` = `PROFBIT` + `IRCALCBIT` + `IROBSVBIT`.

Note that we can have `nchan` > 0 and channel data in the header without having either calculated or observed radiances in a file, to specify a set of channels whose radiances are to be calculated later.

RTP Header Fields

field name	short description	data type	units
ptype	profile type	scalar int32	see note [1]
pfields	profile field set	scalar int32	see note [2]
pmin	min plevs value	scalar float32	millibars
pmax	max plevs value	scalar float32	millibars
ngas	number of gases	scalar int32	[0,MAXGAS]
glist	constituent gas list	ngas int32	HITRAN gas ID
gunit [3]	constituent gas units	ngas int32	gas unit code
pltfid	platform ID	scaler int32	platform code
instid	instrument ID	scaler int32	instrument code
nchan	number of channels	scalar int32	count
ichan	channel numbers	nchan int32	[0,MAXCHAN]
vchan	channel center freq.	nchan float32	cm^-1
vcmin	channel set min freq.	scalar float32	cm^-1
vcmax	channel set max freq.	scalar float32	cm^-1
iudef	user-defined array	MAXIUDEF int32	undefined
itype	user-defined integer	scaler int32	undefined

Notes:

[1] ptype values are
1. level profile LEVPRO = 0
2. layer profile LAYPRO = 1
3. AIRS pseudo-layers AIRSLAY = 2

[2] RTP profile fields are organized in five groups
1. profile data PROFBIT = 1
2. calculated IR radiances IRCALCBIT = 2
3. observed IR radiances IROBSVBIT = 4
For example, a profile with both calculated and observed IR radiances would have pfields = PROFBIT + IRCALCBIT + IROBSVBIT

[3] For suggested gas units code, see the file gas_units_code.txt

Profile Fields -- Surface Data

field name	short description	data type	units
plat	profile latitude	scalar float32	[-90 to 90] deg.
plon	profile longitude	scalar float32	[-180 to 360] deg.
ptime	profile time	scalar float64	TAI
stemp	surface temperature	scalar float32	Kelvins
salti	surface altitude	scalar float32	meters
spres	surface pressure	scalar float32	millibars
landfrac	land fraction	scalar float32	[0 to 1]
landtype	land type code	scalar int32	land code
wspeed	wind speed	scalar float32	meters/sec
nemis [1]	number of emis. pts	scalar int32	[0,MAXEMIS]
efreq [1]	emissivity freq's	nemis float32	cm^-1
emis	surface emissivity	nemis float32	[0 to 1]
rho	surface reflectance	nemis float32	[0 to 1]

Notes:

[1] The nemis and efreq data is also used with cloud emis and rho.

Profile Fields -- Atmospheric Data

field name	short description	data type	units
nlevs	number of press lev's	scalar int32	[0,MAXLEV]
plevs	pressure levels	nlevs float32	millibars
palts	level altitudes	nlevs float32	meters
ptemp	temperature profile	nlevs float32	Kelvins
gas_<i> [1]	gas amount	nlevs float32	HEAD.gunit
gtotal	total column gas amount	ngas float32	undefined
gxover	gas crossover press	ngas float32	millibars
txover	temp crossover press	scalar float32	millibars
co2ppm	CO2 mixing ratio	scalar float32	PPMV

Notes:

[1] There is one field here for each constituent in a file; the constituents are listed in the header field glist. The Fortran and C APIs presents this data as [ngas x nlevs] array gamnt.

Profile Fields -- Cloud Data

field name		short description	data type	units
clrflag		clear flag	scalar int32	[0,1] or clear code
tcc		total cloud cover	scalar float32	[0 to 1]
cc		cloud cover	nlevs float32	[0 to 1]
ciwc		cloud ice water	nlevs float32	g/g
clwc		cloud liquid water	nlevs float32	g/g
ctype	[1]	cloud type code	scalar int32	cloud code
cfrac	[2]	cloud fraction	scalar float32	[0 to 1]
cemis	[2]	cloud top emissivity	nemis float32	[0 to 1]
crho	[2]	cloud top reflectance	nemis float32	[0 to 1]
cprtop	[2]	cloud top pressure	scalar float32	millibars
cprbot		cloud bottom pressure	scalar float32	millibars
cngwat		cloud non-gas water	scalar float32	g/m^2
cpsize		cloud particle size	scalar float32	microns
cstemp	[2]	cloud surface temp	scalar float32	Kelvins
ctype2	[1]	cloud2 type code	scalar int32	cloud code
cfrac2	[2]	cloud2 fraction	scalar float32	[0 to 1]
cemis2	[2]	cloud2 top emissivity	nemis float32	[0 to 1]
crho2	[2]	cloud2 top reflectance	nemis float32	[0 to 1]
cprtop2	[2]	cloud2 top pressure	scalar float32	millibars
cprbot2		cloud2 bottom pressure	scalar float32	millibars
cngwat2		cloud2 non-gas water	scalar float32	g/m^2
cpsize2		cloud2 particle size	scalar float32	microns
cstemp2	[2]	cloud2 surface temp	scalar float32	Kelvins
cfrac12		cloud1+2 fraction	scalar float32	[0 to 1]

Notes:

[1] For suggested cloud type codes see file cloud_code.txt
[2] These cloud fields may instead be used for alternate surfaces.

Profile Fields -- Orientation Data

field name	short description	data type	units
pobs	observer pressure	scalar float32	millibars
zobs	observer height	scalar float32	meters
upwell	radiation direction	scalar int32	1=up, 2=down
scanang	IR scan/view angle	scalar float32	[-90 to 90] deg.
satzen	IR zenith angle	scalar float32	[0 to 180] deg.
satazi	IR azimuth angle	scalar float32	[-180 to 180] deg.
solzen	sun zenith angle	scalar float32	[0 to 180] deg.
solazi	sun azimuth angle	scalar float32	[-180 to 180] deg.
sundist	sun-Earth distance	scalar float32	meters
glint	glint distance	scalar float32	meters

Profile Fields -- Radiance Data

field name	short description	data type	units
rlat	radiance obs lat.	scalar float32	[-90 to 90] deg.
rlon	radiance obs lon.	scalar float32	[-180 to 360] deg.
rtime	radiance obs time	scalar float64	TAI
findex	file (granule) index	scalar int32	index
atrack	along-track index	scalar int32	index
xtrack	cross-track index	scalar int32	index
ifov	field of view index	scalar int32	index
robs1	observed IR rad.	nchan float32	mW/m^2/cm^-1/str
calflag	calibration flag	nchan uint8	see text
robsqual	radiance quality	scalar int32	undefined
freqcal	frequency calibration	scalar float32	undefined
rcalc	calculated IR rad.	nchan float32	mW/m^2/cm^-1/str

Profile Fields -- User Defined Data

field name	short description	data type	units
pnote	profile annotation	MAXPNOTE uint8	text or undefined
udef	user-defined array	MAXUDEF float32	undefined
iudef	user-defined array	MAXIUDEF int32	undefined
itype	user-defined integer	scalar int32	undefined

2.5 HDF Attributes

Attributes are associated either with the header or with the profile record set, and have three parts: the field the attribute is associated with, the attribute name, and the attribute text. In addition to proper field names, the field name "header" is used for general header attributes, and "profile" for general profile attributes.

RTP attributes should typically include such information as title, author, date, and at least a brief descriptive comment. This general information should be set as attributes of the header record. Note that the Fortran/C API uses the 2D `gamnt` array for constituents; this is not actually a Vdata field, and so can not take an attribute. Attributes may be attached to individual constituents with their `gas_<i>` names, where `<i>` is the HITRAN gas ID.

3 Application Interfaces

3.1 The Fortran and C API

The Fortran API consists of four routines: `rtpopen`, `rtpread`, `rtpwrite`, and `rtpclose`. Documentation for these is included in an appendix. The Fortran API uses static structures whose fields, with a few exceptions noted below, are the same as the RTP fields defined above. Normally, only a subset of the Fortran structure fields will be written, with the header field `pfields` and the header size fields used to determine what actually goes into a file. When reading data, if a file contains header or profile fields not in the Fortran structure definition, they are simply ignored. Fields that are defined in the Fortran structure but are not in a file are returned as "BAD", or with the first element BAD, for vectors, while missing size fields are returned as zero.

Attributes are passed to and from the Fortran API in the `RTPATTR` structure array. The records in this array have three fields: `fname`, the field name the attribute is to be associated with, `aname`, the attribute name, and `atext`, the attribute text. The header attribute field name should be either "header", for a general attribute or comment, or a particular header field name. Similarly, the attribute profile field name should be either "profiles" or a specific profile field. Attribute strings need to be null-terminated, with char(0), and the record after the last valid record in an attribute set should have fname set to char(0). See ftest1.f for and ftest2.f examples of reading, writing, and updating attributes.

The Fortran structures differer from the Vdata fields in two ways. First, instead of a `gas_<i>` profile field for each constituent, the Fortran API uses a single array `gamnt(MAXLEV,MAXGAS)` to pass constituent amounts; the `gas_<i>` fields from the HDF file are the columns of this array. Second, the Fortran/C RTP header structure includes the following max size fields, which are not actually written to the Vdata header.

```
mlevs     max number of levels   scalar int32   [0,MAXLEV]
memis     max num of emis pts    scalar int32   [0,MAXEMIS]
```

On a read, these fields are set to the associated profile Vdata field sizes. On a write, they are used to to set the size of the associated Vdata profile fields. They can simply be set to the MAX limits, or to zero if the fields are not used; but using an actual max for the profile set, particularly for mlevs, can give a significant space savings.

A Makefile is supplied to build the RTP API routines as a library file librtp.a, along with some C and Fortran test programs and utilities.

3.2 The Matlab API

The RTP Matlab implementation is a fairly direct mapping between Matlab structure arrays and HDF 4 Vdatas. A read will only return those fields that are in the HDF Vdata, and a write will only write the fields in the Matlab structure. The Matlab RTP API is available as part of the ASL package h4tools; see the README file there for more information. The key routines are rtpread.m and rtpwrite.m. These Matlab functions are reasonably efficient, but there is a key difference between the Matlab and Fortran/C interfaces—the Fortran/C interface loops on records, with only minimal buffering needed, while the Matlab interface loads the entire file in memory, for both reads and writes.

4 Fortran API

```
NAME

  rtpopen -- Fortran interface to open RTP files

SUMMARY

  rtpopen() is used to open an HDF RTP ("Radiative Transfer Profile")
  file for reading or writing profile data.  In addition, it reads or
  writes RTP header data and HDF header and profile attributes.

FORTRAN PARAMETERS

  data type             name      short description       direction
  ---------             -----     -----------------       ---------
  CHARACTER *(*)        fname     RTP file name              IN
  CHARACTER *(*)        mode      'c'=create, 'r'=read       IN
  STRUCTURE /RTPHEAD/   head      RTP header structure       IN/OUT
  STRUCTURE /RTPATTR/   hfatt     RTP header attributes      IN/OUT
  STRUCTURE /RTPATTR/   pfatt     RTP profile attributes     IN/OUT
  INTEGER               rchan     RTP profile channel        OUT

VALUE RETURNED

  0 if successful, -1 on errors

INCLUDE FILES

  rtpdefs.f -- Fortran header, profile, and attribute structures

DISCUSSION

  The valid open modes are 'r' to read an existing file and 'c' to
  create a new file.

  HDF attributes are read and written in an array of RTPATTR
  structures, with one structure record per attribute.  Attributes
  should be terminated with char(0), and are returned that way, for
  a read.  The end of the attribute array is flagged with a char(0)
  at the beginning of the fname field.
```

NAME

rtpread -- Fortran interface to read an RTP profile

SUMMARY

rtpread reads a profile from an open RTP channel, and returns the data in the RTPPROF structure. Successive calls to rtpread return successive profiles from the file, with -1 returned on EOF.

FORTRAN PARAMETERS

data type	name	short description	direction
INTEGER	rchan	RTP profile channel	IN
STRUCTURE /RTPPROF/	prof	RTP profile structure	OUT

VALUE RETURNED

1 (the number of profiles read) on success , -1 on errors or EOF

NAME

rtpwrite -- Fortran interface to write an RTP profile

SUMMARY

rtpwrite writes an RTP profile, represented as the contents of an RTPPROF structure, to an open RTP channel. Successive calls write successive profiles.

FORTRAN PARAMETERS

data type	name	short description	direction
INTEGER	rchan	RTP profile channel	IN
STRUCTURE /RTPPROF/	prof	RTP profile structure	IN

VALUE RETURNED

0 on success, -1 on errors

NAME

rtpclose -- Fortran interface to close an RTP open channel

SUMMARY

rtpclose finishes up after reading or writing an RTP file, writing out any buffers and closing the HDF interface

FORTRAN PARAMETERS

data type	name	short description	direction
INTEGER	rchan	RTP profile channel	IN

VALUE RETURNED

0 on success, -1 on errors

NAME

rtpinit -- initialze RTP profile and header structures

SUMMARY

rtpinit initializes RTP profile structures with some sensible default vaules, and is used when creating a new profile set; it should generally not be used when modifying existing profiles.

rtpinit sets all field sizes to zero, and all data values to "BAD", so that only actual values and sizes need to be written

FORTRAN PARAMETERS

data type	name	short description	direction
STRUCTURE /RTPHEAD/	head	RTP header structure	OUT
STRUCTURE /RTPPROF/	prof	RTP profile structure	OUT

VALUE RETURNED

rtpinit always returns 0

5 Modification

The RTP parameters and field sets can be modified, but some caution is in order. Parameters and structures are defined in rtp.h, in the include directory. The corresponding Fortran declarations are in rtpdefs.f. Declarations that reserve space (initialized structures) are in rtpdef.h. These must all be in agreement. So for example we would set the parameter MAXLEV to 120 in both rtp.h and rtpdefs.f.

The C/Fortran API uses C and Fortran structs as an interface to the RTP files. This is convenient, but the structs are static, with a fixed set of fields, while the format of an RTP file can vary; for example with different sets of fields, or different field sizes. To make this work the interface uses both a static, conventional record structure and a list of field names as strings, with field size and data type. We call the latter data structure FLISTs. The string names are used to match fields in the file with positions in the static buffer structure. The C interface structures are defined in rtp.h, the corresponding Fortran structures in rtpdefs.f, and the associated FLISTs in rtpdef.h. For example, the field `glist` might appear in rtp.h, in the structure rtp_head, as

```
int32       glist[MAXGAS];   /* constituent gas list */
```

The corresponding line in rtpdefs.f, in the Fortran structure /RTPHEAD/, would be

```
integer*4   glist(MAXGAS)    ! constituent gas list
```

`glist` also needs to be set in rtpdef.h, in the struct FLIST hfield, with the line

```
"glist",     DFNT_INT32,     MAXGAS,
```

The positions of the fields should be the same in the C and Fortran structs, the FLISTs, and in the documentation. The parameters NHFIELD and NPFIELD (the number of header and profile fields) in rtp.h should be updated to reflect added or deleted fields. Finally, fields whose size is set with the header values `memis`, `mlevs`, `ngas`, `nchan`, and `pfields` require modification of rtpwrite1.c; look for similar cases there as a pattern.

6 Gas IDs

HITRAN Gas ID List

Gases from the 2008 HITRAN line database

1 = H2O (water vapor)	25 = C2H2 (acetylene)
2 = CO2	26 = C2H2 (ethane)
3 = O3 (ozone)	27 = PH3
4 = N2O	28 = PH3
5 = CO	29 = COF2
6 = CH4 (methane)	30 = SF6
7 = O2 (oxygen)	31 = H2S
8 = NO	32 = HCOOH
9 = SO2	33 = HO2
10 = NO2	34 = O
11 = NH3 (ammonia)	35 = ClONO2 (also see 61)
12 = HNO3 (nitric acid)	36 = NO+
13 = OH	37 = HOBr
14 = HF	38 = C2H4
15 = HCl	39 = CH3OH
16 = HBr	40 = CH3Br
17 = HI	41 = CH3CN
18 = ClO	42 = CF4 (also see 54)
19 = OCS	
20 = H2CO	
21 = HOCl	
22 = N2 (nitrogen)	
23 = HCN	
24 = CH3Cl	
25 = H2O2	

Non-standard Gas ID Lists

Gases represented by cross-sections

51 = CCl3F (CFC-11)
52 = CCl2F2 (CFC-12)
53 = CClF3 (CFC-13)
54 = CF4 (CFC-14)
55 = CHCl2F (CFC-21)
56 = CHClF2 (CFC-22)
57 = C2Cl3F3 (CFC-113)
58 = C2Cl2F4 (CFC-114)
59 = C2ClF5 (CFC-115)
60 = CCl4
61 = ClONO2 (also see 35)
62 = N2O5
63 = HNO4
64 = C2F6
65 = CHCl2CF3 (HCFC-123)
66 = CHClFCF3 (HCFC-124)
67 = CH3CCl2F (HCFC-141b)
68 = CH3CClF2 (HCFC-142b)
69 = CHCl2CF2CF3 (HCFC-225ca)
70 = CClF2CF2CHClF (HCFC-225cb)
71 = CH2F2 (HFC-32)
72 = CHF2CF3 (HFC-134a)
73 = CF3CH3 (HFC-143a)
74 = CH3CHF2 (HFC-152a)

Special purpose IDs
101 self-broadened H2O continuum
102 foreign-broadened H2O continuum
201 Cloud one
202 Cloud two
203 Cloud three