

Joint Center for Satellite Data Assimilation

CRTM: v1.2 User Guide

January, 2009

Change History

Date	Author	Change
2009-01-30	P.van Delst	Initial release.

Contents

1	Introduction	1
1.1	Conventions	1
1.1.1	Naming of Structure Types and Instances of Structures	1
1.1.2	Naming of Definition Modules	1
1.1.3	Naming of Application Modules	2
1.1.4	Naming of I/O Modules	2
1.2	Components	2
1.2.1	Atmospheric Optics	3
1.2.2	Surface Optics	3
1.2.3	Radiative Transfer Solution	3
1.3	Models	3
1.4	Design Framework	4
2	How to obtain the CRTM	6
3	How to build the CRTM library	7
3.1	Build Files	7
3.2	Predefined Build Targets	7
3.3	Building the library	8
3.4	Installing the library	8
3.5	Linking to the library	8
4	How to use the CRTM library	9
4.1	Step by Step Guide	9
4.1.1	Step 1: Access the CRTM module	9
4.1.2	Step 2: Declare the CRTM structures	9
4.1.3	Step 3: Initialise the CRTM	10
4.1.4	Step 4: Allocate the CRTM structures	11
4.1.5	Step 6: Fill the CRTM input structures with data	11
4.1.6	Step 7: Call the required CRTM function	12
4.1.7	Step 8: Destroy the CRTM and cleanup	13

4.2	Interface Descriptions	13
4.2.1	CRTM_Init interface	13
4.2.2	CRTM_Forward interface	16
4.2.3	CRTM_Tangent_Linear interface	18
4.2.4	CRTM_Adjoint interface	20
4.2.5	CRTM_K_Matrix interface	23
4.2.6	CRTM_Destroy interface	26
A	Structure and procedure interface definitions	28
A.1	ChannelInfo Structure	29
A.1.1	CRTM_Associated_ChannelInfo interface	29
A.1.2	CRTM_Allocate_ChannelInfo interface	30
A.1.3	CRTM_Destroy_ChannelInfo interface	31
A.1.4	CRTM_Assign_ChannelInfo interface	32
A.1.5	CRTM_Equal_ChannelInfo interface	33
A.1.6	CRTM_nChannels_ChannelInfo interface	35
A.1.7	CRTM_RCS_ID_ChannelInfo interface	35
A.2	Atmosphere Structure	36
A.2.1	CRTM_Associated_Atmosphere interface	38
A.2.2	CRTM_Allocate_Atmosphere interface	39
A.2.3	CRTM_Destroy_Atmosphere interface	41
A.2.4	CRTM_Assign_Atmosphere interface	42
A.2.5	CRTM_Equal_Atmosphere interface	43
A.2.6	CRTM_SetLayers_Atmosphere interface	45
A.2.7	CRTM_Sum_Atmosphere interface	46
A.2.8	CRTM_Zero_Atmosphere interface	48
A.2.9	CRTM_RCS_ID_Atmosphere interface	49
A.2.10	CRTM_Inquire_Atmosphere_Binary interface	49
A.2.11	CRTM_Read_Atmosphere_Binary interface	50
A.2.12	CRTM_Write_Atmosphere_Binary interface	52
A.3	Cloud Structure	55
A.3.1	CRTM_Associated_Cloud interface	56
A.3.2	CRTM_Allocate_Cloud interface	56
A.3.3	CRTM_Destroy_Cloud interface	58
A.3.4	CRTM_Assign_Cloud interface	59
A.3.5	CRTM_Equal_Cloud interface	60
A.3.6	CRTM_SetLayers_Cloud interface	61
A.3.7	CRTM_Sum_Cloud interface	62
A.3.8	CRTM_Zero_Cloud interface	64
A.3.9	CRTM_RCS_ID_Cloud interface	65
A.3.10	CRTM_Inquire_Cloud_Binary interface	65

A.3.11	CRTM_Read_Cloud_Binary interface	66
A.3.12	CRTM_Write_Cloud_Binary interface	68
A.4	Aerosol Structure	71
A.4.1	CRTM_Associated_Aerosol interface	72
A.4.2	CRTM_Allocate_Aerosol interface	72
A.4.3	CRTM_Destroy_Aerosol interface	74
A.4.4	CRTM_Assign_Aerosol interface	75
A.4.5	CRTM_Equal_Aerosol interface	76
A.4.6	CRTM_SetLayers_Aerosol interface	77
A.4.7	CRTM_Sum_Aerosol interface	79
A.4.8	CRTM_Zero_Aerosol interface	80
A.4.9	CRTM_RCS_ID_Aerosol interface	81
A.4.10	CRTM_Inquire_Aerosol_Binary interface	81
A.4.11	CRTM_Read_Aerosol_Binary interface	82
A.4.12	CRTM_Write_Aerosol_Binary interface	85
A.5	Surface Structure	87
A.5.1	CRTM_Allocate_Surface interface	92
A.5.2	CRTM_Destroy_Surface interface	93
A.5.3	CRTM_Assign_Surface interface	94
A.5.4	CRTM_Equal_Surface interface	95
A.5.5	CRTM_Sum_Surface interface	97
A.5.6	CRTM_Zero_Surface interface	99
A.5.7	CRTM_RCS_ID_Surface interface	99
A.5.8	CRTM_Inquire_Surface_Binary interface	100
A.5.9	CRTM_Read_Surface_Binary interface	101
A.5.10	CRTM_Write_Surface_Binary interface	103
A.5.11	SensorData Structure	106
A.5.12	CRTM_Associated_SensorData interface	107
A.5.13	CRTM_Allocate_SensorData interface	107
A.5.14	CRTM_Destroy_SensorData interface	109
A.5.15	CRTM_Assign_SensorData interface	110
A.5.16	CRTM_Equal_SensorData interface	111
A.5.17	CRTM_RCS_ID_SensorData interface	112
A.6	GeometryInfo Structure	114
A.6.1	CRTM_Compute_GeometryInfo interface	119
A.6.2	CRTM_RCS_ID_GeometryInfo interface	120
A.7	RTSolution Structure	121
A.7.1	CRTM_Associated_RTSolution interface	123
A.7.2	CRTM_Allocate_RTSolution interface	124
A.7.3	CRTM_Destroy_RTSolution interface	125

A.7.4	CRTM_Assign_RTSolution interface	126
A.7.5	CRTM_Equal_RTSolution interface	127
A.7.6	CRTM_RCS_ID_RTSolution interface	129
A.7.7	CRTM_Inquire_RTSolution_Binary interface	129
A.7.8	CRTM_Read_RTSolution_Binary interface	131
A.7.9	CRTM_Write_RTSolution_Binary interface	132
A.8	Options Structure	135
A.8.1	CRTM_Associated_Options interface	136
A.8.2	CRTM_Allocate_Options interface	137
A.8.3	CRTM_Destroy_Options interface	138
A.8.4	CRTM_Assign_Options interface	139
A.8.5	CRTM_Equal_Options interface	140
A.8.6	CRTM_RCS_ID_Options interface	141
B	Valid Sensor Identifiers	143
B.1	Infrared instruments	144
B.2	Microwave instruments	146
C	Utility Software Description	147
C.1	Type_Kinds Module	148
C.2	File_Utility Module	149
C.3	Message_Handler Module	150

List of Figures

1.1	Flowchart of the CRTM Forward and K-Matrix models.	5
2.1	The CRTM ftp site contents (as on Jan.29, 2009)	6
A.1	CRTM_ChannelInfo_type structure definition.	29
A.2	CRTM_Atmosphere_type structure definition.	36
A.3	CRTM_Cloud_type structure definition.	55
A.4	CRTM_Aerosol_type structure definition.	71
A.5	CRTM_Surface_type structure definition.	87
A.6	CRTM_SensorData_type structure definition.	106
A.7	CRTM_GeometryInfo_type structure definition.	114
A.8	Definition of GeometryInfo sensor scan angle component.	116
A.9	Definition of GeometryInfo sensor zenith angle component.	116
A.10	Definition of GeometryInfo sensor azimuth angle component.	117
A.11	Definition of GeometryInfo source zenith angle component.	117
A.12	Definition of GeometryInfo source azimuth angle component.	118
A.13	CRTM_RTSolution_type structure definition.	121
A.14	CRTM_Options_type structure definition.	135

List of Tables

3.1	Predefined makefile targets for CRTM library build. ([†] Untested.)	7
A.1	CRTM <code>Atmosphere</code> structure valid <code>Climatology</code> definitions. The same set as defined for LBLRTM is used.	36
A.2	CRTM <code>Atmosphere</code> structure valid <code>Absorber_ID</code> definitions. The same molecule set as defined for HITRAN is used.	37
A.3	CRTM <code>Atmosphere</code> structure valid <code>Absorber_Units</code> definitions. The same set as defined for LBLRTM is used.	37
A.4	CRTM <code>Cloud</code> structure valid <code>Type</code> definitions.	55
A.5	CRTM <code>Aerosol</code> structure valid <code>Type</code> definitions.	71
A.6	CRTM <code>Surface</code> structure component description.	88
A.7	CRTM <code>Surface</code> structure default values.	89
A.8	CRTM <code>Surface</code> structure valid <code>Land_Type</code> definitions.	90
A.9	CRTM <code>Surface</code> structure valid <code>Water_Type</code> definitions.	90
A.10	CRTM <code>Surface</code> structure valid <code>Snow_Type</code> definitions.	90
A.11	CRTM <code>Surface</code> structure valid <code>Ice_Type</code> definitions.	91
A.12	CRTM <code>SensorData</code> structure component description.	106
A.13	Description of CRTM <code>GeometryInfo</code> structure components defined by the user.	115
A.14	Description of CRTM <code>GeometryInfo</code> structure components derived from user inputs.	115
A.15	CRTM <code>RTSolution</code> structure component description	122
A.16	CRTM <code>Options</code> structure component description	135
B.1	CRTM sensor identifiers for infrared instruments.	145
B.2	CRTM sensor identifiers for microwave instruments.	146

1.1 Conventions

The following are conventions that have been adhered to in the current release of the CRTM framework. They are guidelines intended to make understanding the code at a glance easier, to provide a recognisable “look and feel”, and to minimise name space clashes.

1.1.1 Naming of Structure Types and Instances of Structures

The derived data type, or structure¹ type, naming convention adopted for use in the CRTM is,

[CRTM_] *name_type*

where *name* is an identifier that indicates for what a structure is to be used. All structure type names are suffixed with “_type” and CRTM-specific structure types are prefixed with “CRTM_”. Some examples are,

```
SpcCoeff_type
CRTM_Atmosphere_type
CRTM_RTSolution_type
```

An instance of a structure is then referred to via its *name*, or some sort of derivate of its *name*. Some structure declarations examples are,

```
TYPE(SpcCoeff_type)      :: SpcCoeff
TYPE(CRTM_Atmosphere_type) :: atm, atm_K
TYPE(CRTM_RTSolution_type) :: rts, rts_K
```

where the K-matrix structure variables are identified with a “_K” suffix. Similarly, tangent-linear and adjoint variables are suffixed with “_TL” or “_AD” respectively.

1.1.2 Naming of Definition Modules

Modules containing structure type definitions are termed *definition modules*. These modules contain the actual structure definitions as well as various utility procedures to allocate, destroy, copy etc. structures of the designated type. The naming convention adopted for definition modules in the CRTM is,

[CRTM_] *name_Define*

where, as with the structure type names, all definition module names are suffixed with “_Define” and CRTM-specific definition modules are prefixed with “CRTM_”. Some examples are,

¹The terms “derived type” and “structure” are used interchangeably in this document.

SpcCoeff_Define
CRTM_Atmosphere_Define
CRTM_RTSolution_Define

The actual source code files for these modules have the same name with a “.f90” suffix.

1.1.3 Naming of Application Modules

Modules containing the routines that perform the calculations for the various components of the CRTM are termed *application modules*. The naming convention adopted for application modules in the CRTM is,

CRTM_name

Some examples are,

CRTM_AtmAbsorption
CRTM_SfcOptics
CRTM_RTSolution

However, in this case, *name* does not necessarily refer just to a structure type. Separate application modules are used as required to split up tasks in manageable (and easily maintained) chunks. For example, separate modules have been provided to contain the cloud and aerosol optical property retrieval; similarly separate modules handle different surface types for different instrument types in computing surface optics.

Again, the actual source code files for these modules have the same name with a “.f90” suffix. Note that not all definition modules have a corresponding application module since some structures (e.g. SpcCoeff structures) are simply data containers.

1.1.4 Naming of I/O Modules

Modules containing routines that read and write data from and to files are, naturally, termed I/O modules. Not all data structures have associated I/O modules. The naming convention adopted for these modules in the CRTM is,

[CRTM_]name_Binary_IO

or

[CRTM_]name_netCDF_IO

Some examples are,

SpcCoeff_Binary_IO
SpcCoeff_netCDF_IO
CRTM_Atmosphere_Binary_IO
CRTM_RTSolution_Binary_IO

As with the other module types, the actual source code files for these modules have the same name with a “.f90” suffix.

In the context of the CRTM, the term “Binary” is a euphemism for sequential, unformatted I/O in Fortran. Additionally, since some data structure constructions are not naturally amenable to fixed length data records (at least, not without significant wastage of space), not all data structures used in the CRTM have both netCDF and Binary I/O modules.

1.2 Components

The CRTM is designed around three broad categories: atmospheric optics, surface optics and radiative transfer.

1.2.1 Atmospheric Optics

(**AtmOptics**) This category includes computation of the absorption by atmospheric gases (**AtmAbsorption**) and scattering and absorption by both clouds (**CloudScatter**) and aerosols (**AerosolScatter**).

The gaseous absorption component computes the optical depth of the absorbing constituents in the atmosphere given the pressure, temperature, water vapour, and ozone concentration² profiles.

The scattering component simply interpolates look-up-tables (LUTs) of optical properties – such as mass extinction coefficient and single scatter albedo – for cloud and aerosol types that are then used in the radiative transfer component. See tables A.4 and A.5 for the valid cloud and aerosol types, respectively, that are valid in the CRTM.

1.2.2 Surface Optics

(**SfcOptics**) This category includes the computation of surface emissivity and reflectivity for four gross surface types (land, water, snow, and ice). Each gross surface type has a specified number of specific surface types associated with it. See tables A.8, A.9, A.10, and A.11 for the land, water, snow, and ice surface types, respectively, that are valid in the CRTM.

The CRTM utilises separate models for each gross surface type for each spectral type (infrared and microwave). These models can be either physical models or database/LUT type of models.

1.2.3 Radiative Transfer Solution

(**RTSolution**) This category takes the **AtmOptics** and **SfcOptics** data and solves the radiative transfer problem either clear or scattering atmospheres.

1.3 Models

The CRTM is composed of four models: a forward model, a tangent-linear model, an adjoint model, and a K-matrix model. These can be represented as shown in equations 1.1a to 1.1d.

$$\mathbf{T}_B, \mathbf{R} = \mathbf{F}(\mathbf{T}, \mathbf{q}, T_s, \dots) \quad (1.1a)$$

$$\delta \mathbf{T}_B, \delta \mathbf{R} = \mathbf{H}(\mathbf{T}, \mathbf{q}, T_s, \dots, \delta \mathbf{T}, \delta \mathbf{q}, \delta T_s, \dots) \quad (1.1b)$$

$$\delta^* \mathbf{T}, \delta^* \mathbf{q}, \delta^* T_s, \dots = \mathbf{H}^T(\mathbf{T}, \mathbf{q}, T_s, \dots, \delta^* \mathbf{T}_B) \quad (1.1c)$$

$$\delta^* \mathbf{T}_l, \delta^* \mathbf{q}_l, \delta^* T_{s,l}, \dots = \mathbf{K}(\mathbf{T}, \mathbf{q}, T_s, \dots, \delta^* \mathbf{T}_B) \text{ for } l = 1, 2, \dots, L \quad (1.1d)$$

Here \mathbf{F} is the forward operator that, given the atmospheric temperature and absorber profiles (\mathbf{T} and \mathbf{q}), surface temperature (T_s), etc., produces a vector of channel brightness temperatures (\mathbf{T}_B) and radiances (\mathbf{R}).

The tangent-linear operator, \mathbf{H} , represents a linearisation of the forward model about \mathbf{T} , \mathbf{q} , T_s , etc. and when also supplied with perturbations about the linearisation point (quantities represented by the δ 's) produces the expected perturbations to the brightness temperature and channel radiances.

The adjoint operator, \mathbf{H}^T , is simply the transpose of the tangent-linear operator and produces gradients (the quantities represented by the δ^* 's). It is worth noting that, in the CRTM, these adjoint gradients are accumulated over channel and thus do not represent channel-specific Jacobians.

²Additional trace gas absorption capabilities are being added.

The K-matrix operator³, \mathbf{K} , is effectively the same as the adjoint but with the results preserved by channel (indicated via the subscript l). In the CRTM, the adjoint and K-matrix results are related by,

$$\delta^*x = \sum_{l=1}^L \delta^*x_l \quad (1.2)$$

Thus, the K-matrix results are the derivatives of the diagnostic variables with respect to the prognostic variables, e.g.

$$\delta^*x_l = \frac{\partial T_{B,l}}{\partial x} \quad (1.3)$$

Typically, only the forward or K-matrix models are used in applications. However, the intermediate models are generated and retained for maintenance and testing purposes. Any changes to the CRTM forward model are translated to the tangent-linear model and the latter tested against the former. When the tangent-linear model changes have been verified, the changes then translated to the adjoint model and, as before, the latter is tested against the former. This process is repeated for the adjoint-to-K-matrix models also.

1.4 Design Framework

This document is not really the place to fully discuss the design framework of the CRTM, so it will only be briefly mentioned here. Where appropriate, different physical processes are isolated into their own modules. The CRTM interfaces presented to the user are, at their core, simply drivers for the individual parts. This is shown schematically in the forward and K-matrix model flowcharts of figure 1.1.

A fundamental tenet of the CRTM design is that each component define its own structure definition and application modules to facilitate independent development of an algorithm outside of the mainline CRTM development. By isolating different processes, we can more easily identify requirements for an algorithm with a view to minimise or eliminate potential software conflicts and/or redundancies. The end result sought via this approach is that components developed by different groups can more easily be added into the framework leading to faster implementation of new science and algorithms.

³The term K-matrix is used because references to this operation in the literature commonly use the symbol \mathbf{K}

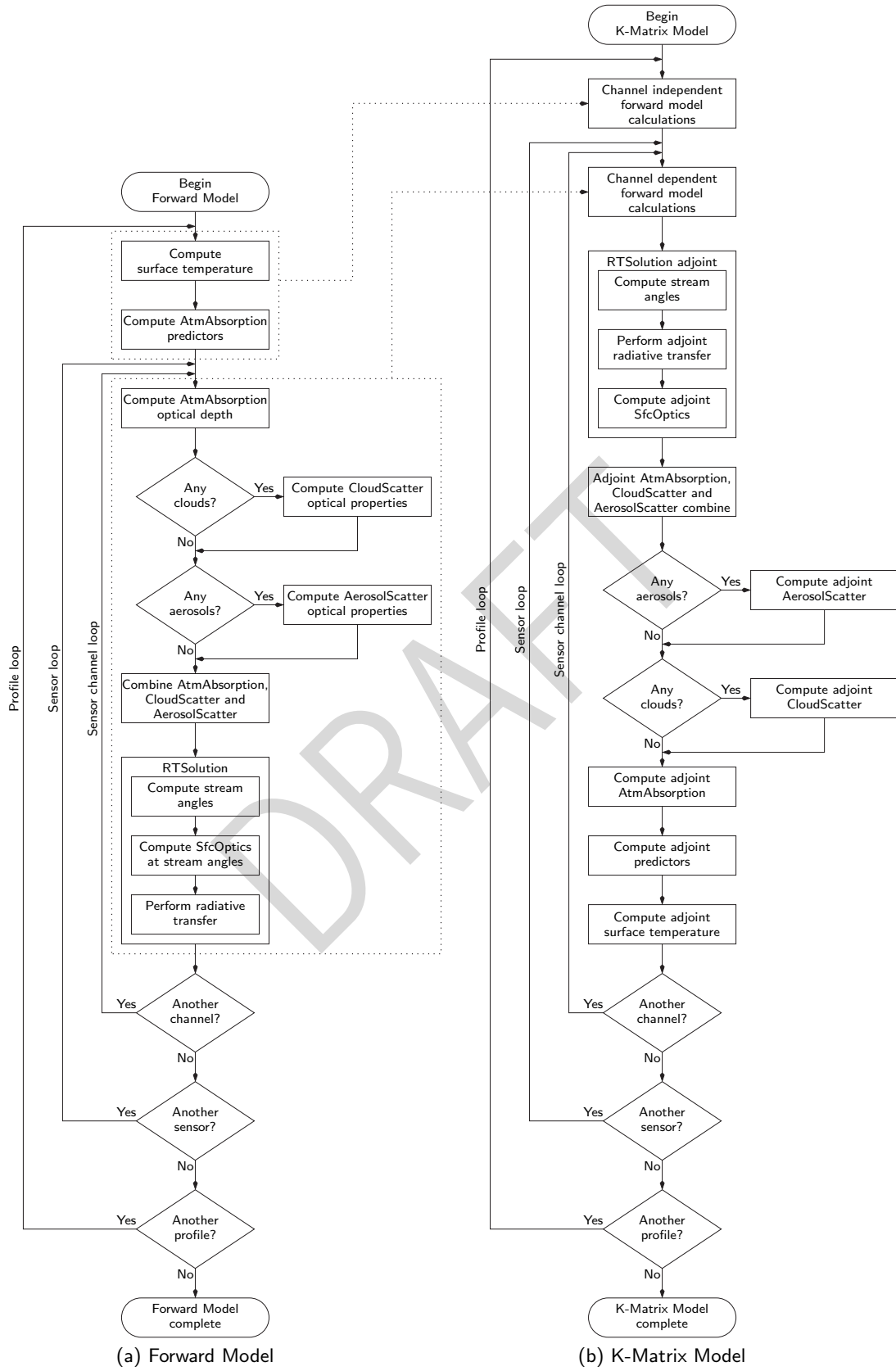


Figure 1.1: Flowchart of the CRTM Forward and K-Matrix models.

2

How to obtain the CRTM

The CRTM source code and coefficients are released as compressed tarballs¹ via the CRTM ftp site:

`ftp://ftp.emc.ncep.noaa.gov/jcsda/CRTM/`

A snapshot of that site is shown in figure 2.1. Note that the source code and coefficients can be released separately as updates to either one can occur independently of the other. Also note that additional releases, e.g. beta or experimental branches, are also made available on this ftp site.

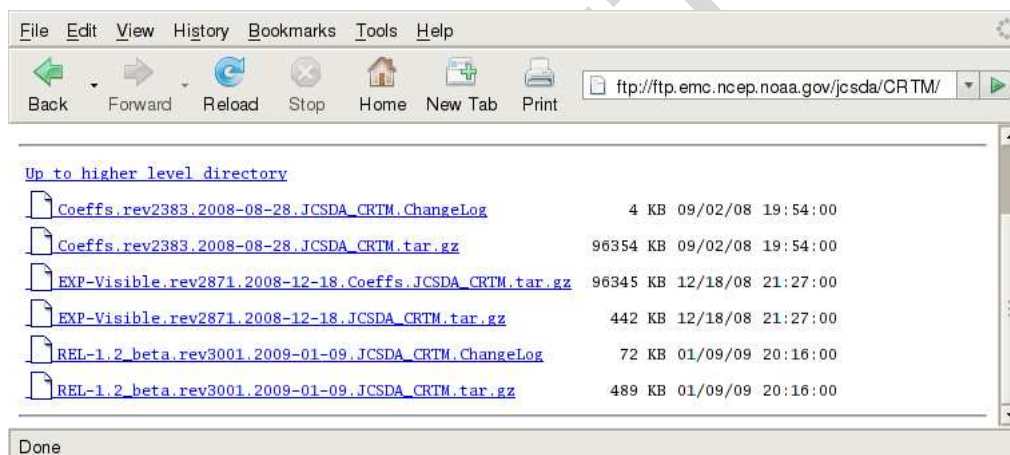


Figure 2.1: The CRTM ftp site contents (as on Jan.29, 2009)

¹A compressed (e.g. gzip'd) tape archive (tar) file.

How to build the CRTM library

3.1 Build Files

The build system for the CRTM is currently quite unsophisticated. It consists of a number of make and include files in the CRTM tarball hierarchy:

makefile : The main makefile
make.macros : The include file containing all the defined macros, including the compiler and linker flags.
make.rules : The include file containing the suffix rules for compiling Fortran95 source code.

3.2 Predefined Build Targets

Several targets are available for specific compilers. The compilers and makefile target names are shown in table 3.1. Both “production” and debug target names are supplied, with the former using compiler switches to produce fast code and the latter using compiler switches to turn on all the available debugging capabilities. Note that the debug targets will produce executables much slower than the production builds.

Platform	Compiler	Production Target	Debug Target
Linux	GNU gfortran	gfortran	gfortran_debug
	Lahey lf95	lahey	lahey_debug
	Intel ifort	intel	intel_debug
	PGI pgf95	pgi	pgi_debug
	g95	g95	g95_debug
	Absoft [†]	absoft	absoft_debug
IBM	AIX xlf95	ibm	ibm_debug
HP	HP-UX f90 [†]	hp	hp_debug
SGI	IRIX f90 [†]	sgi	sgi_debug
Sun	Solaris f90 [†]	sun	sun_debug

Table 3.1: Predefined makefile targets for CRTM library build. ([†] Untested.)

The actual compiler and switch definitions are listed in the **make.macros** file. Also note that not all of the above compilers have been tested for the current release.

3.3 Building the library

The default build is performed by simply typing,

```
make
```

and is operating system sensitive in that the compiler is selected based on the platform. For example, if you build on an IBM/AIX system, the xlf95 compiler is invoked. Doing the same on a linux or MacOSX system invokes the default linux compiler; currently this is the GNU gfortran f95 compiler.

So, if you are using a linux or MacOSX system and want to modify the *default* CRTM build (different compiler, different switches), you will need to edit the `make.macros` file. At the bottom of this file, you will see the following macro definitions,

```
LINUX_FLAGS = $(LINUX_FLAGS_GFORTRAN)
#LINUX_FLAGS = $(LINUX_FLAGS_LAHEY)
#LINUX_FLAGS = $(LINUX_FLAGS_PGI)
#LINUX_FLAGS = $(LINUX_FLAGS_INTEL)
#LINUX_FLAGS = $(LINUX_FLAGS_G95)
```

where you can see the generic `LINUX_FLAGS` macros is set to `LINUX_FLAGS_GFORTRAN`. To use any of the other defined compilers (Lahey, PGI, Intel, or g95), simply uncomment the required line. For example, if your compiler of choice is Intel `ifort` and you want that to be the default build, the change would simply be

```
#LINUX_FLAGS = $(LINUX_FLAGS_GFORTRAN)
#LINUX_FLAGS = $(LINUX_FLAGS_LAHEY)
#LINUX_FLAGS = $(LINUX_FLAGS_PGI)
LINUX_FLAGS = $(LINUX_FLAGS_INTEL)
#LINUX_FLAGS = $(LINUX_FLAGS_G95)
```

Note, however, than an Intel `ifort` build can also be obtained by using the predefined makefile targets as shown in table 3.1.

3.4 Installing the library

An very simple install target is specified in the supplied makefile to put all the necessary include files (the generated `*.mod` files containing all the procedure interface information) in an `/include` subdirectory and the library itself (the generated `libCRTM.a` file) in a `/lib` subdirectory. The make command is

```
make install
```

The `/include` and `/lib` subdirectories can then be moved to a more suitable location on your system, for example: `$HOME/local/CRTM`

3.5 Linking to the library

Let's assume you've built the CRTM library and placed the `/include` and `/lib` subdirectories in your own local area, `$HOME/local/CRTM`. In the makefile for your application that uses the CRTM, you will need to add

```
-I$HOME/local/CRTM/include
```

to your list of compilation switches, and the following to your list of link switches,

```
-L$HOME/local/CRTM/lib -lCRTM
```


How to use the CRTM library

4.1 Step by Step Guide

This section will hopefully get you started using the CRTM library as quickly as possible. Refer to the following sections for more information about the structures and interfaces.

The examples shown here assume you are processing one sensor at a time. The CRTM can handle multiple sensors at once, but specifying the input information in a simple way is difficult; e.g. the **GeometryInfo** structure that is used to specify the sensor viewing geometry – even sensors on the same platform typically have different numbers of fields-of-view (FOVs) per scan. For multiple sensor processing, we'll assume they will be separately processed in parallel.

Because there are many variations in what information is known ahead of time (and by “ahead of time” we mean at compile-time of your code), let's approach this via examples for a fixed number of atmospheric profiles, and a known sensors. It is left as an exercise to the reader to tailor calls to the CRTM in their application code according to their particular needs.

With regards to sensor identification, the CRTM uses a character string – referred to as the **Sensor_Id** – to distinguish sensors and platforms. The lists of currently supported sensors, along with their associated **Sensor_Id**'s, are shown in appendix **B**.

4.1.1 Step 1: Access the CRTM module

All of the CRTM user procedures, parameters, and derived data type definitions are accessible via the container module **CRTM_Module**. Thus, one needs to put the following statement in any calling program, module or procedure,

```
USE CRTM_Module
```

Once you become familiar with the components of the CRTM you require, you can also specify an **ONLY** clause with the **USE** statement,

```
USE CRTM_Module[, ONLY:only-list]
```

where *only-list* is a list of the symbols you want to “import” from **CRTM_Module**. This latter form is the preferred style for self-documenting your code; e.g. when you give the code to someone else, they will be able to identify from which module various symbols in your code originate.

4.1.2 Step 2: Declare the CRTM structures

To compute satellite radiances you need to declare structures for the following information,

1. Atmospheric profile data such as pressure, temperature, absorber amounts, clouds, aerosols, etc. Handled using the **Atmosphere** structure.

2. Surface data such as type of surface, temperature, surface type specific parameters etc. Handled using the `Surface` structure.
3. Geometry information such as sensor scan angle, zenith angle, etc. Handled using the `GeometryInfo` structure.
4. Instrument information, particularly which instrument(s), or sensor(s)¹, you want to simulate. Handled using the `ChannelInfo` structure.
5. Results of the radiative transfer calculation. Handled using the `RTSolution` structure.
6. Optional inputs. Handled using the `Options` structure.

Let's assume you want to process, say, 50 profiles for the NOAA-18 AMSU-A sensor which has 15 channels. The forward model declarations would look something like,

```
! Processing parameters
INTEGER      , PARAMETER :: N_SENSORS  =  1
INTEGER      , PARAMETER :: N_CHANNELS = 15
INTEGER      , PARAMETER :: N_PROFILES = 50
CHARACTER(*) , PARAMETER :: SID(N_SENSORS) = ('amsua_n18')
TYPE(CRTM_ChannelInfo_type) :: chInfo(N_SENSORS)
TYPE(CRTM_GeometryInfo_type) :: gInfo(N_PROFILES)
! Forward declarations
TYPE(CRTM_Atmosphere_type)   :: atm(N_PROFILES)
TYPE(CRTM_Surface_type)     :: sfc(N_PROFILES)
TYPE(CRTM_RTSolution_type)   :: rts(N_CHANNELS,N_PROFILES)
```

If you are also interested in calling the K-matrix model, you will also need the following declarations,

```
! K-Matrix declarations
TYPE(CRTM_Atmosphere_type) :: atm_K(N_CHANNELS,N_PROFILES)
TYPE(CRTM_Surface_type)   :: sfc_K(N_CHANNELS,N_PROFILES)
TYPE(CRTM_RTSolution_type) :: rts_K(N_CHANNELS,N_PROFILES)
```

4.1.3 Step 3: Initialise the CRTM

The CRTM is initialised by calling the `CRTM_Init()` function. This loads all the various coefficient data used by CRTM components into memory for later use. We'll assume that all the required datafiles reside in the subdirectory `./coeff_data` and follow on from the example of Step 2. The CRTM initialisation is profile independent, so we're only dealing with sensor information here. The CRTM initialisation function call looks like,

```
INTEGER :: errStatus
....
errStatus = CRTM_Init( chInfo, Sensor_Id=SID, File_Path='./coeff_data' )
IF ( errStatus /= SUCCESS ) THEN
    handle error...
END IF
```

Here we see for the first time how the CRTM functions let you know if they were successful. As you can see the `CRTM_Init()` function result is an error status that is checked against a parameterised integer error code, `SUCCESS`. The function result should *not* be tested against the actual value of the error code, just its parameterised name. Other available error code parameters are `FAILURE`, `WARNING`, and `INFORMATION` – although the latter is never used as a function result.

¹The terms “instrument” and “sensor” are used interchangeably in this document.

4.1.4 Step 4: Allocate the CRTM structures

Now we need to allocate the *internal* components of the various CRTM structures where necessary to hold the input or output data. In this case, functions are used to perform these “internal” allocations. The function naming convention is `CRTM_Allocate_name` where, for typical usage, the CRTM structures that need to be allocated are the **Atmosphere**, **RTSolution** and, if used, **Options** structures. Potentially, the **SensorData** component of the **Surface** structure may also need to be allocated to allow for input of sensor observations for some of the NESDIS microwave surface emissivity models.

Allocation of the Atmosphere structures

First, we’ll allocate the atmosphere structures to the required dimensions. The forward variable is allocated like so:

```
! Allocate the forward atmosphere structure
errStatus = CRTM_Allocate_Atmosphere( n_Layers    , & ! Input
                                       N_ABSORBERS, & ! Input (always 2)
                                       n_Clouds    , & ! Input
                                       n_Aerosols   , & ! Input
                                       atm          ) ! Output

IF ( errStatus /= SUCCESS ) THEN
    handle error...
END IF
```

and the K-matrix variable is allocated by looping over all profiles,

```
! Allocate the K-matrix atmosphere structure
DO m = 1, N_PROFILES
    errStatus = CRTM_Allocate_Atmosphere( n_Layers    , & ! Input
                                           N_ABSORBERS, & ! Input (always 2)
                                           n_Clouds    , & ! Input
                                           n_Aerosols   , & ! Input
                                           atm_k(:,m)    ) ! Output

    IF ( errStatus /= SUCCESS ) THEN
        handle error...
    END IF
END DO
```

Note that the number of absorbers is in all capitals. In the CRTM, this style convention indicates a parameter. A parameter is used for the number of absorbers in the current CRTM release because the number of absorbers is fixed at two: water vapour and ozone. Future CRTM releases will allow more flexibility in selecting the number of absorbers, but currently the number must be set to two.

4.1.5 Step 6: Fill the CRTM input structures with data

This step simply entails filling the input `atm`, `sfc` and `gInfo` structures with the required information. However, there are some issues that need to be mentioned:

- In the **Atmosphere** structure, the **Climatology** component is not yet used.
- In the **Atmosphere** structure, *both* the level and layer pressure profiles must be specified.
- In the **Atmosphere** structure, the absorber profile data units *must* be mass mixing ratio for water vapour and ppmv for ozone. The **Absorber_Units** component is not yet utilised to allow conversion of different user-supplied concentration units.

- In the **Surface** structure, the sum of the coverage types *must* add up to 1.0.
- In the **GeometryInfo** structure, the sensor zenith and sensor scan angles should be consistent.
- Graphical definitions of the **GeometryInfo** structure sensor scan, sensor zenith, sensor azimuth, source zenith, and source azimuth angles are shown in figures A.8, A.9, A.10, A.11, and A.12 respectively.

For the K-matrix structures, you should zero the K-matrix *outputs*, atm_K, sfc_K,

```
! Zero the K-matrix OUTPUT structures
CALL CRTM_Zero_Atmosphere( atm_K )
CALL CRTM_Zero_Surface( sfc_K )
```

and initialise the K-matrix *input*, rts_K, to provide you with the derivatives you want. For example, if you want the atm_K, sfc_K outputs to contain brightness temperature derivatives, you should initialise rts_K like so,

```
! Initialise the K-Matrix INPUT to provide dTb/dx derivatives
rts_K%Radiance = ZERO
rts_K%Brightness_Temperature = ONE
```

Alternatively, if you want radiance derivatives returned in atm_K and sfc_K, the rts_K structure should be initialised like so,

```
! Initialise the K-Matrix INPUT to provide dR/dx derivatives
rts_K%Radiance = ONE
rts_K%Brightness_Temperature = ZERO
```

For computational efficiency, there is minimal input data checking done so the GIGO² principle applies.

4.1.6 Step 7: Call the required CRTM function

At this point, much of the preparatory heavy lifting has been done. The CRTM function calls themselves are quite simple. For the forward model we do,

```
Error_Status = CRTM_Forward( atm   , & ! Input
                             sfc   , & ! Input
                             gInfo , & ! Input
                             chInfo, & ! Input
                             rts    ) ! Output
IF ( Error_Status /= SUCCESS ) THEN
    handle error...
END IF
```

and for the K-matrix model, the calling syntax is,

```
Error_Status = CRTM_K_Matrix( atm   , & ! Forward input
                              sfc   , & ! Forward input
                              rts_K , & ! K-matrix input
                              gInfo , & ! Input
                              chInfo, & ! Input
                              atm_K , & ! K-matrix output
                              sfc_K , & ! K-matrix output
                              rts    ) ! Forward output
IF ( Error_Status /= SUCCESS ) THEN
    handle error...
END IF
```

²Garbage In, Garbage Out

Note that the K-matrix model also returns the forward model radiances. The **tangent-linear** and **adjoint** models have similar call structures and will not be shown here.

4.1.7 Step 8: Destroy the CRTM and cleanup

The last step is to cleanup. This involves calling the CRTM destruction function

```
Error_Status = CRTM_Destroy( chInfo )
IF ( Error_Status /= SUCCESS ) THEN
    handle error...
END IF
```

to deallocate all the shared coefficient data, as well as calling the individual structure destroy functions to deallocate as required. For the example here, that entails deallocating the forward and K-matrix **Atmosphere** structures, **atm** and **atm_K**,

```
Error_Status = CRTM_Destroy_Atmosphere(atm_K)
IF ( Error_Status /= SUCCESS ) THEN
    handle error...
END IF

Error_Status = CRTM_Destroy_Atmosphere(atm)
IF ( Error_Status /= SUCCESS ) THEN
    handle error...
END IF
```

4.2 Interface Descriptions

4.2.1 CRTM_Init interface

NAME:

CRTM_Init

PURPOSE:

Function to initialise the CRTM.

CALLING SEQUENCE:

```
Error_Status = CRTM_Init( ChannelInfo
                          Sensor_ID      =Sensor_ID      , &
                          CloudCoeff_File =CloudCoeff_File , &
                          AerosolCoeff_File=AerosolCoeff_File, &
                          EmisCoeff_File  =EmisCoeff_File  , &
                          File_Path       =File_Path       , &
                          Quiet           =Quiet           , &
                          Process_ID      =Process_ID      , &
                          Output_Process_ID=Output_Process_ID, &
                          RCS_Id         =RCS_Id         , &
                          Message_Log     =Message_Log     )
```

OUTPUT ARGUMENTS:

ChannelInfo: ChannelInfo structure array populated based on

the contents of the coefficient files and the user inputs.

UNITS: N/A
TYPE: CRTM_ChannelInfo_type
DIMENSION: Rank-1 (n_Sensors)
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Sensor_ID: List of the sensor IDs (e.g. hirs3_n17, amsua_n18, ssmis_f16, etc) with which the CRTM is to be initialised. These Sensor ID are used to construct the sensor specific SpcCoeff and TauCoeff filenames containing the necessary coefficient data, i.e.
<Sensor_ID>.SpcCoeff.bin
and
<Sensor_ID>.TauCoeff.bin
for each sensor Id in the list. IF this argument is not specified, the default SpcCoeff and TauCoeff filenames are "SpcCoeff.bin" and "TauCoeff.bin" respectively.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Same as output ChannelInfo argument
ATTRIBUTES: INTENT(IN), OPTIONAL

CloudCoeff_File: Name of the CRTM Binary format CloudCoeff file containing the scattering coefficient data. If not specified the default filename is "CloudCoeff.bin".
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

AerosolCoeff_File: Name of the CRTM Binary format AerosolCoeff file containing the aerosol absorption and scattering coefficient data. If not specified the default filename is "AerosolCoeff.bin".
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

EmisCoeff_File: Name of the CRTM Binary format EmisCoeff file containing the IRSSEM coefficient data. If not specified the default filename is "EmisCoeff.bin".
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

File_Path: Character string specifying a file path for the input data files. If not specified, the current directory is the default.
UNITS: N/A

TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Quiet: Set this argument to suppress INFORMATION messages being printed to standard output (or the message log file if the Message_Log optional argument is used.) By default, INFORMATION messages are printed. If QUIET = 0, INFORMATION messages are OUTPUT. If QUIET = 1, INFORMATION messages are SUPPRESSED.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Process_ID: Set this argument to the MPI process ID that this function call is running under. This value is used solely for controlling INFORMATION message output. If MPI is not being used, ignore this argument. This argument is ignored if the Quiet argument is set.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Output_Process_ID: Set this argument to the MPI process ID in which all INFORMATION messages are to be output. If the passed Process_ID value agrees with this value the INFORMATION messages are output. This argument is ignored if the Quiet argument is set.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the CRTM initialisation was successful
 == FAILURE an unrecoverable error occurred.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

SIDE EFFECTS:

All public data arrays accessed by this module and its dependencies are overwritten.

RESTRICTIONS:

If specified, the length of the combined file path and filename strings cannot exceed 2000 characters.

COMMENTS:

Note the INTENT on the output ChannelInfo argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

4.2.2 CRTM_Forward interface

NAME:

CRTM_Forward

PURPOSE:

Function that calculates top-of-atmosphere (TOA) radiances and brightness temperatures for an input atmospheric profile or profile set and user specified satellites/channels.

CALLING SEQUENCE:

```
Error_Status = CRTM_Forward( Atmosphere      , &
                             Surface         , &
                             GeometryInfo    , &
                             ChannelInfo     , &
                             RTSolution      , &
                             Options         =Options , &
                             RCS_Id         =RCS_Id   , &
                             Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Atmosphere: Structure containing the Atmosphere data.
 UNITS: N/A
 TYPE: CRTM_Atmosphere_type
 DIMENSION: Rank-1 (n_Profiles)
 ATTRIBUTES: INTENT(IN)

Surface: Structure containing the Surface data.

UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

GeometryInfo: Structure containing the view geometry information.
 UNITS: N/A
 TYPE: CRTM_GeometryInfo_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

ChannelInfo: Structure returned from the CRTM_Init() function that contains the satellite/sensor channel index information.
 UNITS: N/A
 TYPE: CRTM_ChannelInfo_type
 DIMENSION: Rank-1 (n_Sensors)
 ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:
 RTSolution: Structure containing the solution to the RT equation for the given inputs.
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Rank-2 (n_Channels x n_Profiles)
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:
 Options: Options structure containing the optional arguments for the CRTM.
 UNITS: N/A
 TYPE: CRTM_Options_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:
 RCS_Id: Character string containing the Revision Control System Id field for the module.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(OUT), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the computation was successful
== FAILURE an unrecoverable error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

- The Options optional input structure argument contains spectral information (e.g. emissivity) that must have the same spectral dimensionality (the "L" dimension) as the output RTSolution structure.
- The INTENT on the output RTSolution argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

4.2.3 CRTM_Tangent_Linear interface

NAME:

CRTM_Tangent_Linear

PURPOSE:

Function that calculates tangent-linear top-of-atmosphere (TOA) radiances and brightness temperatures for an input atmospheric profile or profile set and user specified satellites/channels.

CALLING SEQUENCE:

```
Error_Status = CRTM_Tangent_Linear( Atmosphere      , &
                                     Surface          , &
                                     Atmosphere_TL    , &
                                     Surface_TL       , &
                                     GeometryInfo     , &
                                     ChannelInfo      , &
                                     RTSolution       , &
                                     RTSolution_TL    , &
                                     Options    =Options , &
                                     RCS_Id    =RCS_Id   , &
                                     Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Atmosphere: Structure containing the Atmosphere data.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Rank-1 (n_Profiles)
ATTRIBUTES: INTENT(IN)

Surface: Structure containing the Surface data.
UNITS: N/A

TYPE: CRTM_Surface_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

Atmosphere_TL: Structure containing the tangent-linear Atmosphere data.
 UNITS: N/A
 TYPE: CRTM_Atmosphere_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

Surface_TL: Structure containing the tangent-linear Surface data.
 UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

GeometryInfo: Structure containing the view geometry information.
 UNITS: N/A
 TYPE: CRTM_GeometryInfo_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

ChannelInfo: Structure returned from the CRTM_Init() function that contains the satellite/sensor channel index information.
 UNITS: N/A
 TYPE: CRTM_ChannelInfo_type
 DIMENSION: Rank-1 (n_Sensors)
 ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

RTSolution: Structure containing the solution to the RT equation for the given inputs.
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Rank-2 (n_Channels x n_Profiles)
 ATTRIBUTES: INTENT(IN OUT)

RTSolution_TL: Structure containing the solution to the tangent-linear RT equation for the given inputs.
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Rank-2 (n_Channels x n_Profiles)
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Options: Options structure containing the optional forward model arguments for the CRTM.
 UNITS: N/A
 TYPE: CRTM_Options_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.

UNITS: N/A

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.

UNITS: N/A

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(OUT), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the computation was successful
 == FAILURE an unrecoverable error occurred

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

COMMENTS:

- The Options optional input structure arguments contain spectral information (e.g. emissivity) that must have the same spectral dimensionality (the "L" dimension) as the output RTSolution structures.
- The INTENT on the output RTSolution arguments are IN OUT rather than just OUT. This is necessary because the arguments may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

4.2.4 CRTM_Adjoint interface

NAME:

CRTM_Adjoint

PURPOSE:

Function that calculates the adjoint of top-of-atmosphere (TOA) radiances and brightness temperatures for an input atmospheric profile or profile set and user specified satellites/channels.

CALLING SEQUENCE:

```
Error_Status = CRTM_Adjoint( Atmosphere      , &
                             Surface         , &
                             RTSolution_AD   , &
```

```

        GeometryInfo          , &
        ChannelInfo           , &
        Atmosphere_AD         , &
        Surface_AD             , &
        RTSolution             , &
        Options                =Options , &
        RCS_Id                 =RCS_Id  , &
        Message_Log=Message_Log )

```

INPUT ARGUMENTS:

Atmosphere: Structure containing the Atmosphere data.
 UNITS: N/A
 TYPE: CRTM_Atmosphere_type
 DIMENSION: Rank-1 (n_Profiles)
 ATTRIBUTES: INTENT(IN)

Surface: Structure containing the Surface data.
 UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN)

RTSolution_AD: Structure containing the RT solution adjoint inputs.
 **NOTE: On EXIT from this function, the contents of this structure may be modified (e.g. set to zero.)
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Rank-2 (n_Channels x n_Profiles)
 ATTRIBUTES: INTENT(IN OUT)

GeometryInfo: Structure containing the view geometry information.
 UNITS: N/A
 TYPE: CRTM_GeometryInfo_type
 DIMENSION: Same as input Atmosphere argument
 ATTRIBUTES: INTENT(IN)

ChannelInfo: Structure returned from the CRTM_Init() function that contains the satellite/sensor channel index information.
 UNITS: N/A
 TYPE: CRTM_ChannelInfo_type
 DIMENSION: Rank-1 (n_Sensors)
 ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Options: Options structure containing the optional forward model arguments for the CRTM.
 UNITS: N/A
 TYPE: CRTM_Options_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

Atmosphere_AD: Structure containing the adjoint Atmosphere data.
 **NOTE: On ENTRY to this function, the contents of this structure should be defined (e.g. initialized to some value based on the position of this function in the call chain.)
 UNITS: N/A
 TYPE: CRTM_Atmosphere_type
 DIMENSION: Same as input Atmosphere argument
 ATTRIBUTES: INTENT(IN OUT)

Surface_AD: Structure containing the tangent-linear Surface data.
 **NOTE: On ENTRY to this function, the contents of this structure should be defined (e.g. initialized to some value based on the position of this function in the call chain.)
 UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Same as input Atmosphere argument
 ATTRIBUTES: INTENT(IN OUT)

RTSolution: Structure containing the solution to the RT equation for the given inputs.
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Same as input RTSolution_AD argument
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(OUT), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the computation was successful
 == FAILURE an unrecoverable error occurred
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

SIDE EFFECTS:

Note that the input adjoint arguments are modified upon exit, and the output adjoint arguments must be defined upon entry. This is a consequence of the adjoint formulation where, effectively, the chain rule is being used and this function could reside anywhere in the chain of derivative terms.

COMMENTS:

- The Options optional structure arguments contain spectral information (e.g. emissivity) that must have the same spectral dimensionality (the "L" dimension) as the RTSolution structures.
- The INTENT on the output RTSolution, Atmosphere_AD, and Surface_AD arguments are IN OUT rather than just OUT. This is necessary because the arguments should be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

4.2.5 CRTM_K_Matrix interface

NAME:

CRTM_K_Matrix

PURPOSE:

Function that calculates the K-matrix of top-of-atmosphere (TOA) radiances and brightness temperatures for an input atmospheric profile or profile set and user specified satellites/channels.

CALLING SEQUENCE:

```
Error_Status = CRTM_K_Matrix( Atmosphere      , &  
                               Surface         , &  
                               RTSolution_K    , &  
                               GeometryInfo    , &  
                               ChannelInfo     , &  
                               Atmosphere_K    , &  
                               Surface_K       , &  
                               RTSolution      , &  
                               Options    =Options , &  
                               RCS_Id      =RCS_Id  , &  
                               Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Atmosphere:	Structure containing the Atmosphere data.
UNITS:	N/A
TYPE:	CRTM_Atmosphere_type
DIMENSION:	Rank-1 (n_Profiles)
ATTRIBUTES:	INTENT(IN)
Surface:	Structure containing the Surface data.
UNITS:	N/A

TYPE: CRTM_Surface_type
 DIMENSION: Same as input Atmosphere argument.
 ATTRIBUTES: INTENT(IN)

RTSolution_K: Structure containing the RT solution K-matrix inputs.
 **NOTE: On EXIT from this function, the contents of this structure may be modified (e.g. set to zero.)
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Rank-2 (n_Channels x n_Profiles)
 ATTRIBUTES: INTENT(IN OUT)

GeometryInfo: Structure containing the view geometry information.
 UNITS: N/A
 TYPE: CRTM_GeometryInfo_type
 DIMENSION: Same as input Atmosphere argument
 ATTRIBUTES: INTENT(IN)

ChannelInfo: Structure returned from the CRTM_Init() function that contains the satellite/sensor channel index information.
 UNITS: N/A
 TYPE: CRTM_ChannelInfo_type
 DIMENSION: Rank-1 (n_Sensors)
 ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Options: Options structure containing the optional forward model arguments for the CRTM.
 UNITS: N/A
 TYPE: CRTM_Options_type
 DIMENSION: Same as input Atmosphere structure
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

Atmosphere_K: Structure containing the K-matrix Atmosphere data.
 **NOTE: On ENTRY to this function, the contents of this structure should be defined (e.g. initialized to some value based on the position of this function in the call chain.)
 UNITS: N/A
 TYPE: CRTM_Atmosphere_type
 DIMENSION: Same as input RTSolution_K argument

ATTRIBUTES: INTENT(IN OUT)

Surface_K: Structure containing the tangent-linear Surface data.
**NOTE: On ENTRY to this function, the contents of this structure should be defined (e.g. initialized to some value based on the position of this function in the call chain.)
UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Same as input RTSolution_K argument
ATTRIBUTES: INTENT(IN OUT)

RTSolution: Structure containing the solution to the RT equation for the given inputs.
UNITS: N/A
TYPE: CRTM_RTSolution_type
DIMENSION: Same as input RTSolution_K argument
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the computation was successful
== FAILURE an unrecoverable error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

Note that the input K-matrix arguments are modified upon exit, and the output K-matrix arguments must be defined upon entry. This is a consequence of the K-matrix formulation where, effectively, the chain rule is being used and this function could reside anywhere in the chain of derivative terms.

COMMENTS:

- The Options optional structure arguments contain spectral information (e.g. emissivity) that must have the same spectral dimensionality (the "L" dimension) as the RTSolution structures.
- The INTENT on the output RTSolution, Atmosphere_K, and Surface_K, arguments are IN OUT rather than just OUT. This is necessary because the arguments should be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

4.2.6 CRTM_Destroy interface

NAME:

CRTM_Destroy

PURPOSE:

Function to deallocate all the shared data arrays allocated and populated during the CRTM initialization.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy( ChannelInfo      , &
                             Process_ID =Process_ID , &
                             RCS_Id      =RCS_Id      , &
                             Message_Log=Message_Log  )
```

OUTPUT ARGUMENTS:

ChannelInfo: Reinitialized ChannelInfo structure.
UNITS: N/A
TYPE: CRTM_ChannelInfo_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Process_ID: Set this argument to the MPI process ID that this function call is running under. This value is used solely for controlling message output. If MPI is not being used, ignore this argument.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
If == SUCCESS the CRTM deallocations were successful
 == FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

All CRTM shared data arrays and structures are deallocated.

COMMENTS:

Note the INTENT on the output ChannelInfo argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

DRAFT

A

Structure and procedure interface definitions

DRAFT

A.1 ChannelInfo Structure

```
TYPE :: CRTM_ChannelInfo_type
  INTEGER :: n_Allocates = 0
  ! Dimensions
  INTEGER :: n_Channels = 0 ! L dimension
  ! Scalar data
  CHARACTER(STRLEN) :: Sensor_ID      = ' '
  INTEGER           :: WMO_Satellite_ID = INVALID_WMO_SATELLITE_ID
  INTEGER           :: WMO_Sensor_ID   = INVALID_WMO_SENSOR_ID
  INTEGER           :: Sensor_Index    = 0
  ! Array data
  INTEGER, POINTER :: Sensor_Channel(:) => NULL() ! L
  INTEGER, POINTER :: Channel_Index(:)  => NULL() ! L
END TYPE CRTM_ChannelInfo_type
```

Figure A.1: CRTM_ChannelInfo_type structure definition.

A.1.1 CRTM_Associated_ChannelInfo interface

NAME:

CRTM_Associated_ChannelInfo

PURPOSE:

Function to test the association status of the pointer members of a CRTM ChannelInfo structure.

CALLING SEQUENCE:

Association_Status = CRTM_Associated_ChannelInfo(ChannelInfo , &
ANY_Test=Any_Test)

INPUT ARGUMENTS:

ChannelInfo: ChannelInfo structure which is to have its pointer member's association status tested.

UNITS: N/A

TYPE: CRTM_ChannelInfo_type

DIMENSION: Scalar OR Rank-1 array

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the ChannelInfo structure pointer members are associated. The default is to test if ALL the pointer members are associated.

If ANY_Test = 0, test if ALL the pointer members are associated. (DEFAULT)

ANY_Test = 1, test if ANY of the pointer members are associated.

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the ChannelInfo pointer members.

- .TRUE. - if ALL the ChannelInfo pointer members are associated, or if the ANY_Test argument is set and ANY of the ChannelInfo pointer members are associated.
- .FALSE. - some or all of the ChannelInfo pointer members are NOT associated.

UNITS: N/A
 TYPE: LOGICAL
 DIMENSION: Same as input ChannelInfo argument

A.1.2 CRTM_Allocate_ChannelInfo interface

NAME:

CRTM_Allocate_ChannelInfo

PURPOSE:

Function to allocate the pointer members of a CRTM ChannelInfo data structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_ChannelInfo( n_Channels, &
                                           ChannelInfo, &
                                           Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Channels: The number of channels in the ChannelInfo structure. Must be > 0.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar or Rank-1 array
 ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

ChannelInfo: ChannelInfo structure with allocated pointer members

UNITS: N/A
 TYPE: CRTM_ChannelInfo_type
 DIMENSION: Same as input n_Channels argument
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any Messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output Messages to standard output.

UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module. If == SUCCESS the structure pointer allocations were successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to one (1) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output ChannelInfo argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.1.3 CRTM_Destroy_ChannelInfo interface

NAME:

CRTM_Destroy_ChannelInfo

PURPOSE:

Function to re-initialize the scalar and pointer members of a CRTM ChannelInfo data structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy_ChannelInfo( ChannelInfo, &
                                         Message_Log=Message_Log )
```

OUTPUT ARGUMENTS:

ChannelInfo: Re-initialized ChannelInfo structure.
UNITS: N/A
TYPE: CRTM_ChannelInfo_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any Messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output Messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter
is not equal to zero (0) upon exiting this
function. This value is incremented and
decremented for every structure allocation
and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output ChannelInfo argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.1.4 CRTM_Assign_ChannelInfo interface

NAME:

CRTM_Assign_ChannelInfo

PURPOSE:

Function to copy valid CRTM ChannelInfo structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_ChannelInfo( ChannelInfo_in      , &  
                                         ChannelInfo_out     , &  
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

ChannelInfo_in: ChannelInfo structure which is to be copied.
UNITS: N/A
TYPE: CRTM_ChannelInfo_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

ChannelInfo_out: Copy of the input structure, ChannelInfo_in.
UNITS: N/A
TYPE: CRTM_ChannelInfo_type
DIMENSION: Same as ChannelInfo_in argument
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any

Messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output Messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module. If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output ChannelInfo argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.1.5 CRTM_Equal_ChannelInfo interface

NAME:

CRTM_Equal_ChannelInfo

PURPOSE:

Function to test if two ChannelInfo structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_ChannelInfo( ChannelInfo_LHS      , &  
                                       ChannelInfo_RHS      , &  
                                       ULP_Scale =ULP_Scale  , &  
                                       Check_All  =Check_All  , &  
                                       Message_Log=Message_Log )
```

INPUT ARGUMENTS:

ChannelInfo_LHS: ChannelInfo structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.
IF (ChannelInfo_LHS == ChannelInfo_RHS).

UNITS: N/A
TYPE: CRTM_ChannelInfo_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

ChannelInfo_RHS: ChannelInfo structure to be compared to; equivalent to the right-hand side of a lexical comparison, e.g.
IF (ChannelInfo_LHS == ChannelInfo_RHS).

UNITS: N/A
 TYPE: CRTM_ChannelInfo_type
 DIMENSION: Same as ChannelInfo_LHS argument
 ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.
 ** NOTE: This is a hook for future changes and is not used.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Check_All: Set this argument to check ALL the floating point channel data of the ChannelInfo structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in ChannelInfo structures.
 If == 0, Return with FAILURE status as soon as ANY difference is found *DEFAULT*
 == 1, Set FAILURE status if ANY difference is found, but continue to check ALL data.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
 UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the structures were equal
 == FAILURE - an error occurred, or
 - the structures were different.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.1.6 CRTM_nChannels_ChannelInfo interface

NAME:

CRTM_nChannels_ChannelInfo

PURPOSE:

Function to return the number of channels defined in a ChannelInfo structure or structure array

CALLING SEQUENCE:

nChannels = CRTM_nChannels_ChannelInfo(ChannelInfo)

INPUT ARGUMENTS:

ChannelInfo: ChannelInfo structure or structure which is to have its channels counted.

UNITS: N/A

TYPE: CRTM_ChannelInfo_type

DIMENSION: Scalar

or

Rank-1

ATTRIBUTES: INTENT(IN)

FUNCTION RESULT:

nChannels: The number of defined channels in the input argument.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

A.1.7 CRTM_RCS_Id_ChannelInfo interface

NAME:

CRTM_RCS_Id_ChannelInfo

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_ChannelInfo(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.

UNITS: N/A

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(OUT)

A.2 Atmosphere Structure

```

TYPE :: CRTM_Atmosphere_type
  INTEGER :: n_Allocates = 0
  ! Dimension values
  INTEGER :: Max_Layers = 0 ! K dimension
  INTEGER :: n_Layers = 0 ! Kuse dimension
  INTEGER :: n_Absorbers = 0 ! J dimension
  INTEGER :: Max_Clouds = 0 ! Nc dimension
  INTEGER :: n_Clouds = 0 ! NcUse dimension
  INTEGER :: Max_Aerosols = 0 ! Na dimension
  INTEGER :: n_Aerosols = 0 ! NaUse dimension
  ! Number of added layers
  INTEGER :: n_Added_Layers = 0
  ! Climatology model associated with the profile
  INTEGER :: Climatology = INVALID_MODEL
  ! Absorber ID and units
  INTEGER, POINTER :: Absorber_ID(:) => NULL() ! J
  INTEGER, POINTER :: Absorber_Units(:) => NULL() ! J
  ! Profile LEVEL and LAYER quantities
  REAL(fp), POINTER :: Level_Pressure(:) => NULL() ! 0:K
  REAL(fp), POINTER :: Pressure(:) => NULL() ! K
  REAL(fp), POINTER :: Temperature(:) => NULL() ! K
  REAL(fp), POINTER :: Absorber(:, :) => NULL() ! K x J
  ! Clouds associated with each profile
  TYPE(CRTM_Cloud_type), POINTER :: Cloud(:) => NULL() ! Nc
  ! Aerosols associated with each profile
  TYPE(CRTM_Aerosol_type), POINTER :: Aerosol(:) => NULL() ! Na
END TYPE CRTM_Atmosphere_type

```

Figure A.2: CRTM_Atmosphere_type structure definition.

Climatology Type	Parameter
Tropical	TROPICAL
Midlatitude summer	MIDLATITUDE_SUMMER
Midlatitude winter	MIDLATITUDE_WINTER
Subarctic summer	SUBARCTIC_SUMMER
Subarctic winter	SUBARCTIC_WINTER
U.S. Standard Atmosphere	US_STANDARD_ATMOSPHERE

Table A.1: CRTM Atmosphere structure valid Climatology definitions. The same set as defined for LBLRTM is used.

Molecule	Parameter	Molecule	Parameter
H ₂ O	H2O_ID	HI	HI_ID
CO ₂	C02_ID	ClO	C10_ID
O ₃	03_ID	OCS	OCS_ID
N ₂ O	N20_ID	H ₂ CO	H2CO_ID
CO	CO_ID	HOCl	H0Cl_ID
CH ₄	CH4_ID	N ₂	N2_ID
O ₂	02_ID	HCN	HCN_ID
NO	NO_ID	CH ₃ I	CH3I_ID
SO ₂	S02_ID	H ₂ O ₂	H2O2_ID
NO ₂	N02_ID	C ₂ H ₂	C2H2_ID
NH ₃	NH3_ID	C ₂ H ₆	C2H6_ID
HNO ₃	HN03_ID	PH ₃	PH3_ID
OH	OH_ID	COF ₂	COF2_ID
HF	HF_ID	SF ₆	SF6_ID
HCl	HCl_ID	H ₂ S	H2S_ID
HBr	HBr_ID	HCOOH	HCOOH_ID

Table A.2: CRTM Atmosphere structure valid Absorber_ID definitions. The same molecule set as defined for HITRAN is used.

Units	Parameter
Volume mixing ratio, ppmv	VOLUME_MIXING_RATIO_UNITS
Number density, cm ⁻³	NUMBER_DENSITY_UNITS
Mass mixing ratio, g/kg	MASS_MIXING_RATIO_UNITS
Mass density, g.m ⁻³	MASS_DENSITY_UNITS
Partial pressure, hPa	PARTIAL_PRESSURE_UNITS
Dewpoint temperature, K (H₂O ONLY)	DEWPOINT_TEMPERATURE_K_UNITS
Dewpoint temperature, C (H₂O ONLY)	DEWPOINT_TEMPERATURE_C_UNITS
Relative humidity, % (H₂O ONLY)	RELATIVE_HUMIDITY_UNITS
Specific amount, g/g	SPECIFIC_AMOUNT_UNITS
Integrated path, mm	INTEGRATED_PATH_UNITS

Table A.3: CRTM Atmosphere structure valid Absorber_Units definitions. The same set as defined for LBLRTM is used.

A.2.1 CRTM_Associated_Atmosphere interface

NAME:

CRTM_Associated_Atmosphere

PURPOSE:

Function to test the association status of the components of a CRTM Atmosphere structure.

CALLING SEQUENCE:

```
Association_Status = CRTM_Associated_Atmosphere( Atmosphere           , &
                                                  ANY_Test      =Any_Test    , &
                                                  Skip_Cloud   =Skip_Cloud  , &
                                                  Skip_Aerosol=Skip_Aerosol )
```

INPUT ARGUMENTS:

Atmosphere: Structure which is to have its pointer member's association status tested.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Scalar, Rank-1, OR Rank-2 array
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the Atmosphere structure components are associated. The default is to test if ALL the components are associated.
If ANY_Test = 0, test if ALL the components are associated. (DEFAULT)
ANY_Test = 1, test if ANY of the components are associated.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Skip_Cloud: Set this argument to not include the Cloud member in the association test. This is required because a valid Atmosphere structure can be cloud-free.
If Skip_Cloud = 0, the Cloud member association status is tested. (DEFAULT)
Skip_Cloud = 1, the Cloud member association status is NOT tested.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Skip_Aerosol: Set this argument to not include the Aerosol member in the association test. This is required because a valid Atmosphere structure can be

aerosol-free.
 If Skip_Aerosol = 0, the Aerosol member association status is tested. (DEFAULT)
 Skip_Aerosol = 1, the Aerosol member association status is NOT tested.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the Atmosphere components.

.TRUE. - if ALL the Atmosphere components are associated, or if the ANY_Test argument is set and ANY of the Atmosphere pointer members are associated.

.FALSE. - some or all of the Atmosphere pointer members are NOT associated.

UNITS: N/A
 TYPE: LOGICAL
 DIMENSION: Same as input Atmosphere argument

A.2.2 CRTM_Allocate_Atmosphere interface

NAME:

CRTM_Allocate_Atmosphere

PURPOSE:

Function to allocate CRTM Atmosphere data structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_Atmosphere( n_Layers      , &
                                           n_Absorbers   , &
                                           n_Clouds      , &
                                           n_Aerosols    , &
                                           Atmosphere    , &
                                           Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: Number of layers dimension.
 Must be > 0.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar OR Rank-1
 See output Atmosphere dimensionality chart
 ATTRIBUTES: INTENT(IN)

n_Absorbers: Number of absorbers dimension. This will be the same for all elements if the Atmosphere argument is an array.

Must be > 0.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

n_Clouds: Number of clouds dimension of Atmosphere data.
 ** Note: Can be = 0 (i.e. clear sky). **
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar OR Rank-1
 See output Atmosphere dimensionality chart
 ATTRIBUTES: INTENT(IN)

n_Aerosols: Number of aerosol types dimension of Atmosphere data.
 ** Note: Can be = 0 (i.e. no aerosols). **
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar OR Rank-1
 See output Atmosphere dimensionality chart
 ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
 messages will be logged. If not specified, or if an
 error occurs opening the log file, the default action
 is to output messages to standard output.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

Atmosphere: Atmosphere structure with allocated components. The
 following table shows the allowable dimension combinations
 for the calling routine, where M == number of profiles:

Input n_Layers dimension	Input n_Absorbers dimension	Input n_Clouds dimension	Input n_Aerosols dimension	Output Atmosphere dimension
scalar	scalar	scalar	scalar	scalar
scalar	scalar	scalar	scalar	M
scalar	scalar	scalar	M	M
scalar	scalar	M	scalar	M
scalar	scalar	M	M	M
M	scalar	scalar	scalar	M
M	scalar	scalar	M	M
M	scalar	M	scalar	M
M	scalar	M	M	M

Note the number of absorbers cannot vary with the profile.

These multiple interfaces are supplied purely for ease of

use depending on what data is available.

UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Scalar or Rank-1
See chart above.
ATTRIBUTES: INTENT(IN OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter
is not equal to one (1) upon exiting this
function. This value is incremented and
decremented for every structure allocation
and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Atmosphere argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.2.3 CRTM_Destroy_Atmosphere interface

NAME:

CRTM_Destroy_Atmosphere

PURPOSE:

Function to re-initialize CRTM Atmosphere data structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy_Atmosphere( Atmosphere , &  
                                         Message_Log=Message_Log )
```

OUTPUT ARGUMENTS:

Atmosphere: Re-initialized Atmosphere structure. In the context of
the CRTM, rank-1 corresponds to an vector of profiles,
and rank-2 corresponds to an array of channels x profiles.
The latter is used in the K-matrix model.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Scalar, Rank-1, OR Rank-2 array
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or

- the structure internal allocation counter is not equal to zero (0) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Atmosphere argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.2.4 CRTM_Assign_Atmosphere interface

NAME:

CRTM_Assign_Atmosphere

PURPOSE:

Function to copy valid CRTM Atmosphere structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_Atmosphere( Atmosphere_in      , &  
                                         Atmosphere_out    , &  
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Atmosphere_in: Atmosphere structure which is to be copied. In the context of the CRTM, rank-1 corresponds to an vector of profiles, and rank-2 corresponds to an array of channels x profiles. The latter is used in the K-matrix model.

UNITS: N/A
TYPE: CRTM_Atmosphere_type

DIMENSION: Scalar, Rank-1, or Rank-2 array
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Atmosphere_out: Copy of the input structure, Atmosphere_in.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Same as Atmosphere_in
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Atmosphere argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.2.5 CRTM_Equal_Atmosphere interface

NAME:

CRTM_Equal_Atmosphere

PURPOSE:

Function to test if two Atmosphere structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_Atmosphere( Atmosphere_LHS, &  
                                       Atmosphere_RHS, &  
                                       ULP_Scale      =ULP_Scale, &  
                                       Percent_Difference=Percent_Difference, &  
                                       Check_All       =Check_All, &  
                                       Message_Log     =Message_Log, )
```

INPUT ARGUMENTS:

Atmosphere_LHS: Atmosphere structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.

IF (Atmosphere_LHS == Atmosphere_RHS).

In the context of the CRTM, rank-1 corresponds to an vector of profiles, and rank-2 corresponds to an array of channels x profiles. The latter is used in the K-matrix model.

UNITS: N/A

TYPE: CRTM_Atmosphere_type

DIMENSION: Scalar, Rank-1, or Rank-2 array

ATTRIBUTES: INTENT(IN)

Atmosphere_RHS: Atmosphere structure to be compared to; equivalent to right-hand side of a lexical comparison, e.g.

IF (Atmosphere_LHS == Atmosphere_RHS).

UNITS: N/A

TYPE: CRTM_Atmosphere_type

DIMENSION: Same as Atmosphere_LHS

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

Percent_Differnece: Percentage difference value to use in comparing the numbers rather than testing within some numerical limit. The ULP_Scale argument is ignored if this argument is specified.

UNITS: N/A

TYPE: REAL(fp)

DIMENSION: Scalar

ATTRIBUTES: OPTIONAL, INTENT(IN)

Check_All: Set this argument to check ALL the floating point channel data of the Atmosphere structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in Atmosphere structures.

If == 0, Return with FAILURE status as soon as

ANY difference is found *DEFAULT*

== 1, Set FAILURE status if ANY difference is found, but continue to check ALL data.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structures were equal
 == FAILURE - an error occurred, or
 - the structures were different.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.2.6 CRTM_SetLayers_Atmosphere interface

NAME:

CRTM_SetLayers_Atmosphere

PURPOSE:

Function to set the number of layers to use in a CRTM Atmosphere structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_SetLayers_Atmosphere( n_Layers      , &
                                           Atmosphere     , &
                                           Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: The value to set the n_Layers component of the Atmosphere structure, as well as any of its Cloud or Aerosol structure components.

UNITS: N/A
 TYPE: CRTM_Atmosphere_type
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

Atmosphere: Atmosphere structure in which the n_Layers dimension is to be updated.

UNITS: N/A

```

        TYPE:          CRTM_Atmosphere_type
        DIMENSION:     Scalar, Rank-1, or Rank-2 array
        ATTRIBUTES:    INTENT(IN OUT)

OUTPUT ARGUMENTS:
    Atmosphere:        On output, the atmosphere structure with the updated
                        n_Layers dimension.
                        UNITS:          N/A
                        TYPE:          CRTM_Atmosphere_type
                        DIMENSION:     Scalar, Rank-1, or Rank-2 array
                        ATTRIBUTES:    INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:
    Message_Log:        Character string specifying a filename in which any
                        messages will be logged. If not specified, or if an
                        error occurs opening the log file, the default action
                        is to output messages to standard output.
                        UNITS:          None
                        TYPE:          CHARACTER(*)
                        DIMENSION:     Scalar
                        ATTRIBUTES:    INTENT(IN), OPTIONAL

FUNCTION RESULT:
    Error_Status:        The return value is an integer defining the error status.
                        The error codes are defined in the Message_Handler module.
                        If == SUCCESS the layer reset was successful
                        == FAILURE an error occurred
                        UNITS:          N/A
                        TYPE:          INTEGER
                        DIMENSION:     Scalar

SIDE EFFECTS:
    The argument Atmosphere is INTENT(IN OUT) and is modified upon output.

COMMENTS:
    - Note that the n_Layers input is *ALWAYS* scalar. Thus, all Atmosphere
      elements will be set to the same number of layers.

    - If n_Layers <= Atmosphere%Max_Layers, then only the dimension value
      of the structure and any sub-structures are changed.

    - If n_Layers > Atmosphere%Max_Layers, then the entire structure is
      reallocated to the required number of layers. No other dimensions
      of the structure or substructures are altered.

```

A.2.7 CRTM_Sum_Atmosphere interface

```

NAME:
    CRTM_Sum_Atmosphere

PURPOSE:

```

Function to perform a sum of two valid CRTM Atmosphere structures. The summation performed is:

$$A = A + \text{Scale_Factor} * B + \text{Offset}$$

where A and B are the CRTM Atmosphere structures, and Scale_Factor and Offset are optional weighting factors.

CALLING SEQUENCE:

```
Error_Status = CRTM_Sum_Atmosphere( A           , &
                                     B           , &
                                     Scale_Factor=Scale_Factor, &
                                     Offset      =Offset      , &
                                     Message_Log =Message_Log  )
```

INPUT ARGUMENTS:

A: Atmosphere structure that is to be added to.
In the context of the CRTM, rank-1 corresponds to an vector of profiles, and rank-2 corresponds to an array of channels x profiles. The latter is used in the K-matrix model.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Scalar, Rank-1, or Rank-2 array
ATTRIBUTES: INTENT(IN OUT)

B: Atmosphere structure that is to be weighted and added to structure A.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Same as A
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Scale_Factor: The first weighting factor used to scale the contents of the input structure, B.
If not specified, Scale_Factor = 1.0.
UNITS: N/A
TYPE: REAL(fp)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Offset: The second weighting factor used to offset the sum of the input structures.
If not specified, Offset = 0.0.
UNITS: N/A
TYPE: REAL(fp)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)

DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

A: Structure containing the summation result,
A = A + Scale_Factor*B + Offset
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Same as B
ATTRIBUTES: INTENT(IN OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

The argument A is INTENT(IN OUT) and is modified upon output.

A.2.8 CRTM_Zero_Atmosphere interface

NAME:

CRTM_Zero_Atmosphere

PURPOSE:

Subroutine to zero-out various members of a CRTM Atmosphere structure -
both scalar and pointer.

CALLING SEQUENCE:

CALL CRTM_Zero_Atmosphere(Atmosphere)

OUTPUT ARGUMENTS:

Atmosphere: Zeroed out Atmosphere structure.
In the context of the CRTM, rank-1 corresponds to an
vector of profiles, and rank-2 corresponds to an array
of channels x profiles. The latter is used in the
K-matrix model.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Scalar, Rank-1, or Rank-2 array
ATTRIBUTES: INTENT(IN OUT)

COMMENTS:

- No checking of the input structure is performed, so there are no
tests for component association status. This means the Atmosphere
structure must have allocated components upon entry to this

routine.

- The dimension components of the structure are **NOT** set to zero.
- The Absorber_ID and Absorber_Units components are **NOT** zeroed out in this routine.
- Note the INTENT on the output Atmosphere argument is IN OUT rather than just OUT. This is necessary because the argument must be defined upon input.

A.2.9 CRTM_RCS_ID_Atmosphere interface

NAME:

CRTM_RCS_ID_Atmosphere

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_Atmosphere(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control
System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT)

A.2.10 CRTM_Inquire_Atmosphere_Binary interface

NAME:

CRTM_Inquire_Atmosphere_Binary

PURPOSE:

Function to inquire Binary format CRTM Atmosphere structure files.

CALLING SEQUENCE:

Error_Status = CRTM_Inquire_Atmosphere_Binary(Filename , &
n_Channels =n_Channels , &
n_Profiles =n_Profiles , &
RCS_Id =RCS_Id , &
Message_Log=Message_Log)

INPUT ARGUMENTS:

Filename: Character string specifying the name of an

Atmosphere format data file to read.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Channels: The number of spectral channels for which there is data in the file. Note that this value will always be 0 for a profile-only dataset-- it only has meaning for K-matrix data.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

n_Profiles: The number of profiles in the data file.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the version control Id field for the module.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the Binary file inquire was successful
== FAILURE an unrecoverable error occurred.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

A.2.11 CRTM_Read_Atmosphere_Binary interface

NAME:

CRTM_Read_Atmosphere_Binary

PURPOSE:

Function to read Binary format CRTM Atmosphere structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Read_Atmosphere_Binary( Filename           , &
                                             Atmosphere         , &
                                             Quiet             , &
                                             n_Channels        , &
                                             n_Profiles        , &
                                             RCS_Id            , &
                                             Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an
Atmosphere format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Atmosphere: Structure containing the Atmosphere data. Note the
following meanings attributed to the dimensions of
the structure array:
Rank-1: M profiles.
Only profile data are to be read in. The file
does not contain channel information. The
dimension of the structure is understood to
be the PROFILE dimension.
Rank-2: L channels x M profiles
Channel and profile data are to be read in.
The file contains both channel and profile
information. The first dimension of the
structure is the CHANNEL dimension, the second
is the PROFILE dimension. This is to allow
K-matrix structures to be read in with the
same function.
UNITS: N/A
TYPE: CRTM_Atmosphere_type
DIMENSION: Rank-1 (M) or Rank-2 (L x M)
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.) By default, INFORMATION messages are printed.
If QUIET = 0, INFORMATION messages are OUTPUT.
QUIET = 1, INFORMATION messages are SUPPRESSED.
UNITS: N/A
TYPE: INTEGER

DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Channels: The number of channels for which data was read. Note that this value will always be 0 for a profile-only dataset-- it only has meaning for K-matrix data.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

n_Profiles: The number of profiles for which data was read.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the version control Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the Binary file read was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Atmosphere argument is IN OUT rather than just OUT. This is necessary because the argument may be defined on input. To prevent memory leaks, the IN OUT INTENT is a must.

A.2.12 CRTM_Write_Atmosphere_Binary interface

NAME:

CRTM_Write_Atmosphere_Binary

PURPOSE:

Function to write Binary format Atmosphere files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Write_Atmosphere_Binary( Filename           , &
                                             Atmosphere         , &
                                             Quiet              =Quiet      , &
                                             RCS_Id             =RCS_Id       , &
                                             Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an output
Atmosphere format data file.

UNITS: N/A

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN)

Atmosphere: Structure containing the Atmosphere data to write.
Note the following meanings attributed to the
dimensions of the structure array:

Rank-1: M profiles.

Only profile data are to be read in. The file
does not contain channel information. The
dimension of the structure is understood to
be the PROFILE dimension.

Rank-2: L channels x M profiles

Channel and profile data are to be read in.
The file contains both channel and profile
information. The first dimension of the
structure is the CHANNEL dimension, the second
is the PROFILE dimension. This is to allow
K-matrix structures to be read in with the
same function.

UNITS: N/A

TYPE: CRTM_Atmosphere_type

DIMENSION: Rank-1 (M) or Rank-2 (L x M)

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.) By default, INFORMATION messages are printed.
If QUIET = 0, INFORMATION messages are OUTPUT.

QUIET = 1, INFORMATION messages are SUPPRESSED.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the version control Id field for the module.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the Binary file write was successful
== FAILURE an unrecoverable error occurred.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

- If the output file already exists, it is overwritten.
- If an error occurs *during* the write phase, the output file is deleted before returning to the calling routine.

A.3 Cloud Structure

```
TYPE :: CRTM_Cloud_type
  INTEGER :: n_Allocates = 0
  ! Dimension values
  INTEGER :: Max_Layers = 0 ! K dimension.
  INTEGER :: n_Layers = 0 ! Kuse dimension.
  ! Number of added layers
  INTEGER :: n_Added_Layers = 0
  ! Cloud type
  INTEGER :: Type = NO_CLOUD
  ! Cloud state variables
  REAL(fp), POINTER :: Effective_Radius(:) => NULL() ! K. Units are microns
  REAL(fp), POINTER :: Effective_Variance(:) => NULL() ! K. Units are microns^2
  REAL(fp), POINTER :: Water_Content(:) => NULL() ! K. Units are kg/m^2
END TYPE CRTM_Cloud_type
```

Figure A.3: CRTM_Cloud_type structure definition.

Cloud Type	Parameter
Water	WATER_CLOUD
Ice	ICE_CLOUD
Rain	RAIN_CLOUD
Snow	SNOW_CLOUD
Graupel	GRAUPEL_CLOUD
Hail	HAIL_CLOUD

Table A.4: CRTM Cloud structure valid Type definitions.

A.3.1 CRTM_Associated_Cloud interface

NAME:

CRTM_Associated_Cloud

PURPOSE:

Function to test the association status of a CRTM Cloud structure.

CALLING SEQUENCE:

```
Association_Status = CRTM_Associated_Cloud( Cloud           , &
                                           ANY_Test=Any_Test )
```

INPUT ARGUMENTS:

Cloud: Cloud structure which is to have its pointer member's association status tested.

UNITS: N/A

TYPE: CRTM_Cloud_type

DIMENSION: Scalar OR Rank-1 array

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the Cloud structure pointer members are associated. The default is to test if ALL the pointer members are associated.

If ANY_Test = 0, test if ALL the pointer members are associated. (DEFAULT)

ANY_Test = 1, test if ANY of the pointer members are associated.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the Cloud pointer members.

.TRUE. - if ALL the Cloud pointer members are associated, or if the ANY_Test argument is set and ANY of the Cloud pointer members are associated.

.FALSE. - some or all of the Cloud pointer members are NOT associated.

UNITS: N/A

TYPE: LOGICAL

DIMENSION: Same as input Cloud argument

A.3.2 CRTM_Allocate_Cloud interface

NAME:

CRTM_Allocate_Cloud

PURPOSE:

Function to allocate CRTM Cloud structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_Cloud( n_Layers           , &  
                                     Cloud              , &  
                                     Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: Number of layers for which there is cloud data.
Must be > 0.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Cloud: Cloud structure with allocated pointer members.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Same as input n_Layers argument
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter
is not equal to one (1) upon exiting this
function. This value is incremented and
decremented for every structure allocation
and deallocation respectively.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Cloud argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.3.3 CRTM_Destroy_Cloud interface

NAME:

CRTM_Destroy_Cloud

PURPOSE:

Function to re-initialize CRTM Cloud structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy_Cloud( Cloud           , &
                                   Message_Log=Message_Log )
```

OUTPUT ARGUMENTS:

Cloud: Re-initialized Cloud structure.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to zero (0) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Cloud argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.3.4 CRTM_Assign_Cloud interface

NAME:

CRTM_Assign_Cloud

PURPOSE:

Function to copy valid CRTM Cloud structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_Cloud( Cloud_in           , &
                                   Cloud_out           , &
                                   Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Cloud_in: Cloud structure which is to be copied.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Scalar or Rank-1 array
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Cloud_out: Copy of the input structure, Cloud_in.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Same as Cloud_in argument
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Cloud argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.3.5 CRTM_Equal_Cloud interface

NAME:

CRTM_Equal_Cloud

PURPOSE:

Function to test if two Cloud structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_Cloud( Cloud_LHS      , &
                                Cloud_RHS      , &
                                ULP_Scale    =ULP_Scale  , &
                                Check_All    =Check_All   , &
                                Message_Log=Message_Log   )
```

INPUT ARGUMENTS:

Cloud_LHS: Cloud structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.
IF (Cloud_LHS == Cloud_RHS).
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

Cloud_RHS: Cloud structure to be compared to; equivalent to right-hand side of a lexical comparison, e.g.
IF (Cloud_LHS == Cloud_RHS).
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Same as Cloud_RHS argument
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Check_All: Set this argument to check ALL the floating point channel data of the Cloud structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in Cloud structures.
If == 0, Return with FAILURE status as soon as

ANY difference is found *DEFAULT*
 == 1, Set FAILURE status if ANY difference is
 found, but continue to check ALL data.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any
 messages will be logged. If not specified, or if an
 error occurs opening the log file, the default action
 is to output messages to standard output.
 UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
 The error codes are defined in the Message_Handler module.
 If == SUCCESS the structures were equal
 == FAILURE - an error occurred, or
 - the structures were different.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.3.6 CRTM_SetLayers_Cloud interface

NAME:

CRTM_SetLayers_Cloud

PURPOSE:

Function to set the number of layers to use in a CRTM Cloud
 structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_SetLayers_Cloud( n_Layers      , &
                                     Cloud          , &
                                     Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: The value to set the n_Layers component of the
 Cloud structure, as well as those of any of its
 structure components.
 UNITS: N/A
 TYPE: CRTM_Cloud_type
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

Cloud: Cloud structure in which the n_Layers dimension is to be updated.
 UNITS: N/A
 TYPE: CRTM_Cloud_type
 DIMENSION: Scalar OR Rank-1 array
 ATTRIBUTES: INTENT(IN OUT)

OUTPUT ARGUMENTS:

Cloud: On output, the Cloud structure with the updated n_Layers dimension.
 UNITS: N/A
 TYPE: CRTM_Cloud_type
 DIMENSION: Scalar or Rank-1 array
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
 UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the layer reset was successful
 == FAILURE an error occurred
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

SIDE EFFECTS:

The argument Cloud is INTENT(IN OUT) and is modified upon output. The elements of the structure are reinitialised

COMMENTS:

- Note that the n_Layers input is **ALWAYS** scalar. Thus, all Cloud elements will be set to the same number of layers.
- If n_Layers <= Cloud%Max_Layers, then only the dimension value of the structure and any sub-structures are changed.
- If n_Layers > Cloud%Max_Layers, then the entire structure is reallocated to the required number of layers.

A.3.7 CRTM_Sum_Cloud interface

NAME:

CRTM_Sum_Cloud

PURPOSE:

Function to perform a sum of two valid CRTM Cloud structures. The summation performed is:

$$A = A + \text{Scale_Factor} * B + \text{Offset}$$

where A and B are the CRTM Cloud structures, and Scale_Factor and Offset are optional weighting factors.

CALLING SEQUENCE:

```
Error_Status = CRTM_Sum_Cloud( A, &
                                B, &
                                Scale_Factor=Scale_Factor, &
                                Offset=Offset, &
                                Message_Log=Message_Log )
```

INPUT ARGUMENTS:

A: Cloud structure that is to be added to.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

B: Cloud structure that is to be weighted and added to structure A.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Same as A
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

A: Structure containing the summation result,
A = A + Scale_Factor*B + Offset
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Same as B
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Scale_Factor: The first weighting factor used to scale the contents of the input structure, B.
If not specified, Scale_Factor = 1.0.
UNITS: N/A
TYPE: REAL(fp)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Offset: The second weighting factor used to offset the sum of the input structures.
If not specified, Offset = 0.0.
UNITS: N/A
TYPE: REAL(fp)
DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure summation was successful
== FAILURE an error occurred

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

SIDE EFFECTS:

The argument A is INTENT(IN OUT) and is modified upon output.

A.3.8 CRTM_Zero_Cloud interface

NAME:

CRTM_Zero_Cloud

PURPOSE:

Subroutine to zero-out various members of a CRTM Cloud structure - both scalar and pointer.

CALLING SEQUENCE:

CALL CRTM_Zero_Cloud(Cloud)

OUTPUT ARGUMENTS:

Cloud: Zeroed out Cloud structure.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Scalar or Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

COMMENTS:

- No checking of the input structure is performed, so there are no tests for pointer member association status. This means the Cloud structure must have allocated pointer members upon entry to this routine.
- The dimension components of the structure are *NOT* set to zero.

- The cloud type component is **NOT** reset.
- Note the INTENT on the output Cloud argument is IN OUT rather than just OUT. This is necessary because the argument must be defined upon input.

A.3.9 CRTM_RCS_ID_Cloud interface

NAME:

CRTM_RCS_ID_Cloud

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_Cloud(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control
System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT)

A.3.10 CRTM_Inquire_Cloud_Binary interface

NAME:

CRTM_Inquire_Cloud_Binary

PURPOSE:

Function to inquire Binary format CRTM Cloud structure files.

CALLING SEQUENCE:

Error_Status = CRTM_Inquire_Cloud_Binary(Filename , &
n_Clouds =n_Clouds , &
RCS_Id =RCS_Id , &
Message_Log=Message_Log)

INPUT ARGUMENTS:

Filename: Character string specifying the name of a
Cloud format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Clouds: The number of Cloud profiles in the data file.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the Revision Control System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module. If == SUCCESS the Binary file inquire was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

A.3.11 CRTM_Read_Cloud_Binary interface

NAME:

CRTM_Read_Cloud_Binary

PURPOSE:

Function to read Binary format CRTM Cloud structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Read_Cloud_Binary( Filename      , &  
                                       Cloud          , &  
                                       Quiet          =Quiet      , &  
                                       No_File_Close=No_File_Close, &  
                                       No_Allocate   =No_Allocate , &  
                                       n_Clouds      =n_Clouds    , &  
                                       RCS_Id        =RCS_Id      , &  
                                       Message_Log    =Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of a
Cloud format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Cloud: Structure containing the Cloud data.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Rank-1
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.)
If == 0, INFORMATION messages are OUTPUT [DEFAULT].
== 1, INFORMATION messages are SUPPRESSED.
If not specified, information messages are output.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

No_File_Close: Set this argument to NOT close the file upon exit.
If == 0, the input file is closed upon exit [DEFAULT]
== 1, the input file is NOT closed upon exit.
If not specified, the default action is to close the
input file upon exit.
the
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(IN)

No_Allocate: Set this argument to NOT allocate the output Cloud
structure in this routine based on the data dimensions
read from the input data file. This assumes that the
structure has already been allocated prior to calling
this function.
If == 0, the output Cloud structure is allocated [DEFAULT]
== 1, the output Cloud structure is NOT allocated
If not specified, the default action is to allocate
the output Cloud structure to the dimensions specified
in the input data file.
the
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

ATTRIBUTES: OPTIONAL, INTENT(IN)

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Clouds: The actual number of cloud profiles read in.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the Revision Control System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module. If == SUCCESS the Binary file read was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

If an error occurs:
- the input file is closed and,
- the output Cloud structure is deallocated.

COMMENTS:

Note the INTENT on the output Cloud argument is IN OUT rather than just OUT. This is necessary because the argument may be defined on input. To prevent memory leaks, the IN OUT INTENT is a must.

A.3.12 CRTM_Write_Cloud_Binary interface

NAME:

CRTM_Write_Cloud_Binary

PURPOSE:

Function to write Binary format Cloud files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Write_Cloud_Binary( Filename           , &
                                         Cloud              , &
                                         Quiet              , &
                                         =Quiet            , &
                                         No_File_Close=No_File_Close, &
                                         RCS_Id            , &
                                         =RCS_Id           , &
                                         Message_Log      =Message_Log   )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an output
Cloud format data file.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

Cloud: Structure containing the Cloud data.
UNITS: N/A
TYPE: CRTM_Cloud_type
DIMENSION: Rank-1
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.)
If == 0, INFORMATION messages are OUTPUT [DEFAULT].
== 1, INFORMATION messages are SUPPRESSED.
If not specified, information messages are output.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

No_File_Close: Set this argument to NOT close the file upon exit.
If == 0, the input file is closed upon exit [DEFAULT]
== 1, the input file is NOT closed upon exit.
If not specified, the default action is to close the
input file upon exit.
the
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(IN)

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A

TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control
System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the Binary file write was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

- If the output file already exists, it is overwritten.
- If an error occurs during *writing*, the output file is deleted before returning to the calling routine.

A.4 Aerosol Structure

```

TYPE :: CRTM_Aerosol_type
  INTEGER :: n_Allocates = 0
  ! Dimensions
  INTEGER :: Max_Layers = 0 ! K dimension.
  INTEGER :: n_Layers = 0 ! K dimension
  ! Number of added layers
  INTEGER :: n_Added_Layers = 0
  ! Aerosol type
  INTEGER :: Type = NO_AEROSOL
  ! Aerosol state variables
  REAL(fp), POINTER :: Effective_Radius(:) => NULL() ! K. Units are microns
  REAL(fp), POINTER :: Concentration(:) => NULL() ! K. Units are kg/m^2
END TYPE CRTM_Aerosol_type

```

Figure A.4: CRTM_Aerosol.type structure definition.

Aerosol Type	Parameter
Dust	DUST_AEROSOL
Sea salt SSAM ¹	SEASALT_SSAM_AEROSOL
Sea salt SSCM ²	SEASALT_SSCM_AEROSOL
Dry organic carbon	DRY_ORGANIC_CARBON_AEROSOL
Wet organic carbon	WET_ORGANIC_CARBON_AEROSOL
Dry black carbon	DRY_BLACK_CARBON_AEROSOL
Wet black carbon	WET_BLACK_CARBON_AEROSOL
Sulfate	SULFATE_AEROSOL

Table A.5: CRTM Aerosol structure valid Type definitions.

A.4.1 CRTM_Associated_Aerosol interface

NAME:

CRTM_Associated_Aerosol

PURPOSE:

Function to test the association status of the pointer members of a CRTM Aerosol structure.

CALLING SEQUENCE:

```
Association_Status = CRTM_Associated_Aerosol( Aerosol           , &
                                             ANY_Test=Any_Test  )
```

INPUT ARGUMENTS:

Aerosol: Aerosol structure which is to have its pointer member's association status tested.

UNITS: N/A

TYPE: CRTM_Aerosol_type

DIMENSION: Scalar OR Rank-1 array

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the Aerosol structure pointer members are associated. The default is to test if ALL the pointer members are associated.

If ANY_Test = 0, test if ALL the pointer members are associated. (DEFAULT)

ANY_Test = 1, test if ANY of the pointer members are associated.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the Aerosol pointer members.

.TRUE. - if ALL the Aerosol pointer members are associated, or if the ANY_Test argument is set and ANY of the Aerosol pointer members are associated.

.FALSE. - some or all of the Aerosol pointer members are NOT associated.

UNITS: N/A

TYPE: LOGICAL

DIMENSION: Same as input Aerosol argument

A.4.2 CRTM_Allocate_Aerosol interface

NAME:

CRTM_Allocate_Aerosol

PURPOSE:

Function to allocate the pointer members of the CRTM Aerosol data structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_Aerosol( n_Layers           , &
                                       Aerosol             , &
                                       Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: Number of atmospheric layers dimension.
Must be > 0
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

Aerosol: Aerosol structure with allocated pointer members.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Same as input n_Layers argument
ATTRIBUTES: INTENT(IN OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to one (1) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Aerosol argument is IN OUT rather than

just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.4.3 CRTM_Destroy_Aerosol interface

NAME:

CRTM_Destroy_Aerosol

PURPOSE:

Function to re-initialize the scalar and pointer members of a CRTM Aerosol data structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy_Aerosol( Aerosol      , &
                                     Message_Log=Message_Log )
```

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

Aerosol: Re-initialized Aerosol structure.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to zero (0) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Aerosol argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon

input. To prevent memory leaks, the IN OUT INTENT is a must.

A.4.4 CRTM_Assign_Aerosol interface

NAME:

CRTM_Assign_Aerosol

PURPOSE:

Function to copy valid CRTM Aerosol structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_Aerosol( Aerosol_in      , &
                                   Aerosol_out      , &
                                   Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Aerosol_in: Aerosol structure which is to be copied.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Aerosol_out: Copy of the input structure, Aerosol_in.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Same as Aerosol_in argument
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Aerosol argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.4.5 CRTM_Equal_Aerosol interface

NAME:

CRTM_Equal_Aerosol

PURPOSE:

Function to test if two Aerosol structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_Aerosol( Aerosol_LHS      , &
                                   Aerosol_RHS      , &
                                   ULP_Scale =ULP_Scale , &
                                   Check_All  =Check_All  , &
                                   Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Aerosol_LHS: Aerosol structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.
IF (Aerosol_LHS == Aerosol_RHS).
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar OR Rank-1 array
ATTRIBUTES: INTENT(IN)

Aerosol_RHS: Aerosol structure to be compared to; equivalent to right-hand side of a lexical comparison, e.g.
IF (Aerosol_LHS == Aerosol_RHS).
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Same as Aerosol_LHS
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Check_All: Set this argument to check ALL the floating point channel data of the Aerosol structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in Aerosol structures.
 If == 0, Return with FAILURE status as soon as ANY difference is found *DEFAULT*
 == 1, Set FAILURE status if ANY difference is found, but continue to check ALL data.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the structures were equal
 == FAILURE - an error occurred, or
 - the structures were different.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.4.6 CRTM_SetLayers_Aerosol interface

NAME:

CRTM_SetLayers_Aerosol

PURPOSE:

Function to set the number of layers to use in a CRTM Aerosol structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_SetLayers_Aerosol( n_Layers, &
                                         Aerosol, &
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: The value to set the n_Layers component of the

Aerosol structure, as well as those of any of its structure components.

UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

Aerosol: Aerosol structure in which the n_Layers dimension is to be updated.

UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar or Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

OUTPUT ARGUMENTS:

Aerosol: On output, the Aerosol structure with the updated n_Layers dimension.

UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar or Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: None
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the layer reset was successful

== FAILURE an error occurred

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

The argument Aerosol is INTENT(IN OUT) and is modified upon output. The elements of the structure are reinitialised

COMMENTS:

- Note that the n_Layers input is *ALWAYS* scalar. Thus, all Aerosol elements will be set to the same number of layers.
- If n_Layers <= Aerosol%Max_Layers, then only the dimension value of the structure and any sub-structures are changed.
- If n_Layers > Aerosol%Max_Layers, then the entire structure is reallocated to the required number of layers.

A.4.7 CRTM_Sum_Aerosol interface

NAME:

CRTM_Sum_Aerosol

PURPOSE:

Function to perform a sum of two valid CRTM Aerosol structures. The summation performed is:

$$A = A + \text{Scale_Factor} * B + \text{Offset}$$

where A and B are the CRTM Aerosol structures, and Scale_Factor and Offset are optional weighting factors.

CALLING SEQUENCE:

```
Error_Status = CRTM_Sum_Aerosol( A           , &
                                B           , &
                                Scale_Factor=Scale_Factor, &
                                Offset      =Offset      , &
                                Message_Log =Message_Log   )
```

INPUT ARGUMENTS:

A: Aerosol structure that is to be added to.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar OR Rank-1
ATTRIBUTES: INTENT(IN OUT)

B: Aerosol structure that is to be weighted and added to structure A.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Same as A
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Scale_Factor: The first weighting factor used to scale the contents of the input structure, B.
If not specified, Scale_Factor = 1.0.
UNITS: N/A
TYPE: REAL(fp)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Offset: The second weighting factor used to offset the sum of the input structures.
If not specified, Offset = 0.0.
UNITS: N/A
TYPE: REAL(fp)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OUTPUT ARGUMENTS:

A: Structure containing the summation result,
A = A + Scale_Factor*B + Offset
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Same as B
ATTRIBUTES: INTENT(IN OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure summation was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

The argument A is INTENT(IN OUT) and is modified upon output.

A.4.8 CRTM_Zero_Aerosol interface

NAME:

CRTM_Zero_Aerosol

PURPOSE:

Subroutine to zero-out all members of a CRTM Aerosol structure - both scalar and pointer.

CALLING SEQUENCE:

CALL CRTM_Zero_Aerosol(Aerosol)

OUTPUT ARGUMENTS:

Aerosol: Zeroed out Aerosol structure.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Scalar or Rank-1 array
ATTRIBUTES: INTENT(IN OUT)

COMMENTS:

- No checking of the input structure is performed, so there are no tests for pointer member association status. This means the Aerosol structure must have allocated pointer members upon entry to this routine.
- The dimension components of the structure are **NOT** set to zero.
- The aerosol type component is **NOT** reset.
- Note the INTENT on the output Aerosol argument is IN OUT rather than just OUT. This is necessary because the argument must be defined upon input.

A.4.9 CRTM_RCS_ID_Aerosol interface

NAME:

CRTM_RCS_ID_Aerosol

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_Aerosol(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control
System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT)

A.4.10 CRTM_Inquire_Aerosol_Binary interface

NAME:

CRTM_Inquire_Aerosol_Binary

PURPOSE:

Function to inquire Binary format CRTM Aerosol structure files.

CALLING SEQUENCE:

Error_Status = CRTM_Inquire_Aerosol_Binary(Filename , &
n_Aerosols =n_Aerosols, &
RCS_Id =RCS_Id , &
Message_Log=Message_Log)

INPUT ARGUMENTS:

Filename: Character string specifying the name of a
Aerosol format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Aerosols: The number of Aerosol profiles in the data file.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the Revision Control
System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the Binary file inquire was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

A.4.11 CRTM_Read_Aerosol_Binary interface

NAME:

CRTM_Read_Aerosol_Binary

PURPOSE:

Function to read Binary format CRTM Aerosol structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Read_Aerosol_Binary( Filename           , &
                                         Aerosol            , &
                                         Quiet              , &
                                         No_File_Close=No_File_Close, &
                                         No_Allocate        , &
                                         n_Aerosols         , &
                                         RCS_Id             , &
                                         Message_Log        , &
                                         Message_Log        )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of a
Aerosol format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Aerosol: Structure containing the Aerosol data.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Rank-1
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.)
If == 0, INFORMATION messages are OUTPUT [DEFAULT].
== 1, INFORMATION messages are SUPPRESSED.
If not specified, information messages are output.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

No_File_Close: Set this argument to NOT close the file upon exit.
If == 0, the input file is closed upon exit [DEFAULT]
== 1, the input file is NOT closed upon exit.
If not specified, the default action is to close the
input file upon exit.
the
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(IN)

No_Allocate: Set this argument to NOT allocate the output Aerosol
structure in this routine based on the data dimensions
read from the input data file. This assumes that the
structure has already been allocated prior to calling
this function.
If == 0, the output Aerosol structure is allocated [DEFAULT]

== 1, the output Aerosol structure is NOT allocated
If not specified, the default action is to allocate
the output Aerosol structure to the dimensions specified
in the input data file.

the

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(IN)

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Aerosols: The actual number of aerosol profiles read in.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the Revision Control
System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the Binary file read was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

If an error occurs:
- the input file is closed and,
- the output Aerosol structure is deallocated.

COMMENTS:

Note the INTENT on the output Aerosol argument is IN OUT rather
than just OUT. This is necessary because the argument may be defined on
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.4.12 CRTM_Write_Aerosol_Binary interface

NAME:

CRTM_Write_Aerosol_Binary

PURPOSE:

Function to write Binary format Aerosol files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Write_Aerosol_Binary( Filename, &
                                           Aerosol, &
                                           Quiet, =Quiet, &
                                           No_File_Close=No_File_Close, &
                                           RCS_Id, =RCS_Id, &
                                           Message_Log, =Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an output
Aerosol format data file.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

Aerosol: Structure containing the Aerosol data.
UNITS: N/A
TYPE: CRTM_Aerosol_type
DIMENSION: Rank-1
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.)
If == 0, INFORMATION messages are OUTPUT [DEFAULT].
== 1, INFORMATION messages are SUPPRESSED.
If not specified, information messages are output.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

No_File_Close: Set this argument to NOT close the file upon exit.
If == 0, the input file is closed upon exit [DEFAULT]
== 1, the input file is NOT closed upon exit.
If not specified, the default action is to close the
input file upon exit.
the
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(IN)

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module. If == SUCCESS the Binary file write was successful
== FAILURE an unrecoverable error occurred.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

- If the output file already exists, it is overwritten.
- If an error occurs during *writing*, the output file is deleted before returning to the calling routine.

A.5 Surface Structure

```
TYPE :: CRTM_Surface_type
  INTEGER :: n_Allocates = 0
  ! Dimension values
  INTEGER :: Max_Sensors = 0 ! N dimension
  INTEGER :: n_Sensors = 0 ! Nuse dimension
  ! Gross type of surface determined by coverage
  REAL(fp) :: Land_Coverage = ZERO
  REAL(fp) :: Water_Coverage = ZERO
  REAL(fp) :: Snow_Coverage = ZERO
  REAL(fp) :: Ice_Coverage = ZERO
  ! Surface type independent data
  REAL(fp) :: Wind_Speed = DEFAULT_WIND_SPEED
  REAL(fp) :: Wind_Direction = DEFAULT_WIND_DIRECTION
  ! Land surface type data
  INTEGER :: Land_Type = DEFAULT_LAND_TYPE
  REAL(fp) :: Land_Temperature = DEFAULT_LAND_TEMPERATURE
  REAL(fp) :: Soil_Moisture_Content = DEFAULT_SOIL_MOISTURE_CONTENT
  REAL(fp) :: Canopy_Water_Content = DEFAULT_CANOPY_WATER_CONTENT
  REAL(fp) :: Vegetation_Fraction = DEFAULT_VEGETATION_FRACTION
  REAL(fp) :: Soil_Temperature = DEFAULT_SOIL_TEMPERATURE
  ! Water type data
  INTEGER :: Water_Type = DEFAULT_WATER_TYPE
  REAL(fp) :: Water_Temperature = DEFAULT_WATER_TEMPERATURE
  REAL(fp) :: Salinity = DEFAULT_SALINITY
  ! Snow surface type data
  INTEGER :: Snow_Type = DEFAULT_SNOW_TYPE
  REAL(fp) :: Snow_Temperature = DEFAULT_SNOW_TEMPERATURE
  REAL(fp) :: Snow_Depth = DEFAULT_SNOW_DEPTH
  REAL(fp) :: Snow_Density = DEFAULT_SNOW_DENSITY
  REAL(fp) :: Snow_Grain_Size = DEFAULT_SNOW_GRAIN_SIZE
  ! Ice surface type data
  INTEGER :: Ice_Type = DEFAULT_ICE_TYPE
  REAL(fp) :: Ice_Temperature = DEFAULT_ICE_TEMPERATURE
  REAL(fp) :: Ice_Thickness = DEFAULT_ICE_THICKNESS
  REAL(fp) :: Ice_Density = DEFAULT_ICE_DENSITY
  REAL(fp) :: Ice_Roughness = DEFAULT_ICE_ROUGHNESS
  ! SensorData containing channel brightness temperatures
  TYPE(CRTM_SensorData_type) :: SensorData ! N
END TYPE CRTM_Surface_type
```

Figure A.5: CRTM_Surface.type structure definition.

Component	Description	Units	Dimensions
n_Sensors	The number of sensors for which data is provided inside the SensorData structure	N/A	Scalar
Land_Coverage	Fraction of the FOV that is land surface	N/A	Scalar
Water_Coverage	Fraction of the FOV that is water surface	N/A	Scalar
Snow_Coverage	Fraction of the FOV that is snow surface	N/A	Scalar
Ice_Coverage	Fraction of the FOV that is ice surface	N/A	Scalar
Wind_Speed	Surface wind speed	m.s^{-1}	Scalar
Wind_Direction	Surface wind direction	deg. E from N	Scalar
Land_Type	Land surface type	N/A	Scalar
Land_Temperature	Land surface temperature	Kelvin	Scalar
Soil_Moisture_Content	Volumetric water content of the soil	g.cm^{-3}	Scalar
Canopy_Water_Content	Gravimetric water content of the canopy	g.cm^{-3}	Scalar
Vegetation_Fraction	Vegetation fraction of the surface	%	Scalar
Soil_Temperature	Soil temperature	Kelvin	Scalar
Water_Type	Water surface type	N/A	Scalar
Water_Temperature	Water surface temperature	Kelvin	Scalar
Salinity	Water salinity	‰	Scalar
Snow_Type	Snow surface type	N/A	Scalar
Snow_Temperature	Snow surface temperature	Kelvin	Scalar
Snow_Depth	Snow depth	mm	Scalar
Snow_Density	Snow density	g.m^{-3}	Scalar
Snow_Grain_Size	Snow grain size	mm	Scalar
Ice_Type	Ice surface type	N/A	Scalar
Ice_Temperature	Ice surface temperature	Kelvin	Scalar
Ice_Thickness	Thickness of ice	mm	Scalar
Ice_Density	Density of ice	g.m^{-3}	Scalar
Ice_Roughness	Measure of the surface roughness of the ice	N/A	Scalar
SensorData	Satellite sensor data required for some surface emissivity algorithms	N/A	Scalar

Table A.6: CRTM Surface structure component description.

Parameter	Value	Units
Surface type independent data		
DEFAULT_WIND_SPEED	5.0	m.s ⁻¹
DEFAULT_WIND_DIRECTION	0.0	deg. E from N
Land surface type data		
DEFAULT_LAND_TYPE	GRASS_SOIL	N/A
DEFAULT_LAND_TEMPERATURE	283.0	K
DEFAULT_SOIL_MOISTURE_CONTENT	0.05	g.cm ⁻³
DEFAULT_CANOPY_WATER_CONTENT	0.05	g.cm ⁻³
DEFAULT_VEGETATION_FRACTION	0.3	30%
DEFAULT_SOIL_TEMPERATURE	283.0	K
Water type data		
DEFAULT_WATER_TYPE	SEA_WATER	N/A
DEFAULT_WATER_TEMPERATURE	283.0	K
DEFAULT_SALINITY	33.0	ppmv
Snow surface type data		
DEFAULT_SNOW_TYPE	NEW_SNOW	N/A
DEFAULT_SNOW_TEMPERATURE	263.0	K
DEFAULT_SNOW_DEPTH	50.0	mm
DEFAULT_SNOW_DENSITY	0.2	g.cm ⁻³
DEFAULT_SNOW_GRAIN_SIZE	2.0	mm
Ice surface type data		
DEFAULT_ICE_TYPE	FRESH_ICE	N/A
DEFAULT_ICE_TEMPERATURE	263.0	K
DEFAULT_ICE_THICKNESS	10.0	mm
DEFAULT_ICE_DENSITY	0.9	g.cm ⁻³
DEFAULT_ICE_ROUGHNESS	0.0	N/A

Table A.7: CRTM Surface structure default values.

Land Type	Parameter
Compacted soil	COMPACTED_SOIL
Tilled soil	TILLED_SOIL
Sand	SAND
Rock	ROCK
Irrigated low vegetation	IRRIGATED_LOW_VEGETATION
Meadow grass	MEADOW_GRASS
Scrub	SCRUB
Broadleaf forest	BROADLEAF_FOREST
Pine forest	PINE_FOREST
Tundra	TUNDRA
Grass-soil	GRASS_SOIL
Broadleaf-pine forest	BROADLEAF_PINE_FOREST
Grass scrub	GRASS_SCRUB
Soil-grass-scrub	SOIL_GRASS_SCRUB
Urban concrete	URBAN_CONCRETE
Pine brush	PINE_BRUSH
Broadleaf brush	BROADLEAF_BRUSH
Wet soil	WET_SOIL
Scrub-soil	SCRUB_SOIL
Broadleaf(70)-Pine(30)	BROADLEAF70_PINE30

Table A.8: CRTM Surface structure valid Land_Type definitions.

Water Type	Parameter
Sea water	SEA_WATER
Fresh water	FRESH_WATER

Table A.9: CRTM Surface structure valid Water_Type definitions.

Snow Type	Parameter
Wet snow	WET_SNOW
Grass after snow	GRASS_AFTER_SNOW
Powder snow	POWDER_SNOW
RS snow(A)	RS_SNOW_A
RS snow(B)	RS_SNOW_B
RS snow(C)	RS_SNOW_C
RS snow(D)	RS_SNOW_D
RS snow(E)	RS_SNOW_E
Thin Crust snow	THIN_CRUST_SNOW
Thick crust snow	THICK_CRUST_SNOW
Shallow snow	SHALLOW_SNOW
Deep snow	DEEP_SNOW
Crust snow	CRUST_SNOW
Medium snow	MEDIUM_SNOW
Bottom crust snow(A)	BOTTOM_CRUST_SNOW_A
Bottom crust snow(B)	BOTTOM_CRUST_SNOW_B

Table A.10: CRTM Surface structure valid Snow_Type definitions.

Ice Type	Parameter
Fresh ice	FRESH_ICE
First year sea ice	FIRST_YEAR_SEA_ICE
Multiple year sea ice	MULTI_YEAR_SEA_ICE
Ice floe	ICE_FLOE
Ice ridge	ICE_RIDGE

Table A.11: CRTM Surface structure valid Ice_Type definitions.

A.5.1 CRTM_Allocate_Surface interface

NAME:

CRTM_Allocate_Surface

PURPOSE:

Function to allocate CRTM Surface data structures.

NOTE: This function is a wrapper for the CRTM_SensorData allocation routine to provide the functionality and convenience for allocation of both scalar and rank-1 Surface structures in the same manner as for CRTM_Atmosphere_type structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_Surface( n_Channels      , &
                                       Surface           , &
                                       Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Channels: Number of channels dimension of Surface%SensorData structure
** Note: Can be = 0 (i.e. no sensor data). **
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar OR Rank-1
See output Surface dimensionality table
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Surface: Surface structure with allocated SensorData pointer members. The following table shows the allowable dimension combinations for the calling routine, where M == number of profiles/surface locations:

Input	Output
n_Channels	Surface
dimension	dimension
scalar	scalar
scalar	M
M	M

These multiple interfaces are supplied purely for ease of use depending on what data is available.

UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Scalar or Rank-1
See chart above.
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to one (1) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Surface argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.2 CRTM_Destroy_Surface interface

NAME:

CRTM_Destroy_Surface

PURPOSE:

Function to re-initialize the scalar and pointer members of Surface data structures.

NOTE: This function is mostly a wrapper for the CRTM_SensorData destruction routine to provide the functionality and convenience of allocation of both scalar and rank-1 Surface structures in the same manner as for CRTM_Atmosphere_type structures.

CALLING SEQUENCE:

Error_Status = CRTM_Destroy_Surface(Surface , &
Message_Log=Message_Log)

OUTPUT ARGUMENTS:

Surface: Re-initialized Surface structure. In the context of the CRTM, rank-1 corresponds to an vector of profiles, and rank-2 corresponds to an array of channels x profiles.

The latter is used in the K-matrix model.
UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Scalar, Rank-1, OR Rank-2 array
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to zero (0) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Surface argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.3 CRTM_Assign_Surface interface

NAME:

CRTM_Assign_Surface

PURPOSE:

Function to copy valid Surface structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_Surface( Surface_in      , &  
                                   Surface_out      , &  
                                   Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Surface_in: Surface structure which is to be copied. In the context of

the CRTM, rank-1 corresponds to an vector of profiles,
and rank-2 corresponds to an array of channels x profiles.
The latter is used in the K-matrix model.

UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Scalar, Rank-1, OR Rank-2 array
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Surface_out: Copy of the input structure, Surface_in.
UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Same as Surface_in
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Surface argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.4 CRTM_Equal_Surface interface

NAME:

CRTM_Equal_Surface

PURPOSE:

Function to test if two Surface structures are equal.

CALLING SEQUENCE:

Error_Status = CRTM_Equal_Surface(Surface_LHS , &
Surface_RHS , &

```

        ULP_Scale          =ULP_Scale          , &
        Percent_Difference=Percent_Difference, &
        Check_All          =Check_All          , &
        Message_Log        =Message_Log        )

```

INPUT ARGUMENTS:

Surface_LHS: Surface structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.

IF (Surface_LHS == Surface_RHS).

UNITS: N/A

TYPE: CRTM_Surface_type

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN)

Surface_RHS: Surface structure to be compared to; equivalent to the right-hand side of a lexical comparison, e.g.

IF (Surface_LHS == Surface_RHS).

UNITS: N/A

TYPE: CRTM_Surface_type

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

Percent_Difference: Percentage difference value to use in comparing the numbers rather than testing within some numerical limit. The ULP_Scale argument is ignored if this argument is specified.

UNITS: N/A

TYPE: REAL(fp)

DIMENSION: Scalar

ATTRIBUTES: OPTIONAL, INTENT(IN)

Check_All: Set this argument to check ALL the floating point channel data of the Surface structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in Surface structures.

If == 0, Return with FAILURE status as soon as

ANY difference is found *DEFAULT*

== 1, Set FAILURE status if ANY difference is

found, but continue to check ALL data.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
 UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structures were equal
 == FAILURE - an error occurred, or
 - the structures were different.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.5.5 CRTM_Sum_Surface interface

NAME:

CRTM_Sum_Surface

PURPOSE:

Function to perform a sum of two valid CRTM Surface structures. The summation performed is:

$$A = A + \text{Scale_Factor} * B + \text{Offset}$$

where A and B are the CRTM Surface structures, and Scale_Factor and Offset are optional weighting factors.

CALLING SEQUENCE:

```
Error_Status = CRTM_Sum_Surface( A, &
                                B, &
                                Scale_Factor=Scale_Factor, &
                                Offset=Offset, &
                                Message_Log=Message_Log )
```

INPUT ARGUMENTS:

A: Surface structure that is to be added to.
 In the context of the CRTM, rank-1 corresponds to an vector of profiles, and rank-2 corresponds to an array of channels x profiles. The latter is used in the K-matrix model.

UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Scalar, Rank-1, or Rank-2 array
 ATTRIBUTES: INTENT(IN OUT)

B: Surface structure that is to be weighted and added to structure A.
 UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Same as A
 ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

A: Structure containing the weight sum result,
 $A = A + \text{Scale_Factor} * B + \text{Offset}$
 UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Same as B
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Scale_Factor: The first weighting factor used to scale the contents of the input structure, B.
 If not specified, Scale_Factor = 1.0.
 UNITS: N/A
 TYPE: REAL(fp)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Offset: The second weighting factor used to offset the sum of the input structures.
 If not specified, Offset = 0.0.
 UNITS: N/A
 TYPE: REAL(fp)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the structure assignment was successful
 == FAILURE an error occurred
 UNITS: N/A
 TYPE: INTEGER

DIMENSION: Scalar

SIDE EFFECTS:

The argument A is INTENT(IN OUT) and is modified upon output.

COMMENTS:

The SensorData component of the Surface structures are not operated on.

A.5.6 CRTM_Zero_Surface interface

NAME:

CRTM_Zero_Surface

PURPOSE:

Subroutine to zero-out members of a CRTM Surface structure.

CALLING SEQUENCE:

CALL CRTM_Zero_Surface(Surface)

OUTPUT ARGUMENTS:

Surface: Zeroed out Surface structure.
In the context of the CRTM, rank-1 corresponds to an
vector of profiles, and rank-2 corresponds to an array
of channels x profiles. The latter is used in the
K-matrix model.
UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Scalar, Rank-1, or Rank-2 array
ATTRIBUTES: INTENT(IN OUT)

COMMENTS:

- No checking of the input structure is performed.
- The SensorData dimension and structure components are *NOT* reset.
- Note the INTENT on the output Surface argument is IN OUT rather than just OUT. This is necessary because the argument must be defined upon input.

A.5.7 CRTM_RCS_ID_Surface interface

NAME:

CRTM_RCS_ID_Surface

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

```
CALL CRTM_RCS_Id_Surface( RCS_Id )
```

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT)

A.5.8 CRTM_Inquire_Surface_Binary interface

NAME:

CRTM_Inquire_Surface_Binary

PURPOSE:

Function to inquire Binary format CRTM Surface structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Inquire_Surface_Binary( Filename , &  
                                             n_Channels =n_Channels , &  
                                             n_Profiles =n_Profiles , &  
                                             RCS_Id =RCS_Id , &  
                                             Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an Surface format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Channels: The number of spectral channels for which there is data in the file. Note that this value will always be 0 for a profile-only dataset-- it only has meaning for K-matrix data.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(OUT)

n_Profiles: The number of profiles in the data file.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the version control Id
 field for the module.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
 The error codes are defined in the Message_Handler module.
 If == SUCCESS the Binary inquiry was successful
 == FAILURE an unrecoverable error occurred.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.5.9 CRTM_Read_Surface_Binary interface

NAME:

CRTM_Read_Surface_Binary

PURPOSE:

Function to read Binary format CRTM Surface structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Read_Surface_Binary( Filename      , &
                                         Surface        , &
                                         Quiet          =Quiet      , &
                                         n_Channels     =n_Channels , &
                                         n_Profiles     =n_Profiles , &
                                         RCS_Id         =RCS_Id      , &
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an
 Surface format data file to read.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar

ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Surface: Structure containing the Surface data. Note the following meanings attributed to the dimensions of the structure array:
Rank-1: M profiles.
Only profile data are to be read in. The file does not contain channel information. The dimension of the structure is understood to be the PROFILE dimension.
Rank-2: L channels x M profiles
Channel and profile data are to be read in. The file contains both channel and profile information. The first dimension of the structure is the CHANNEL dimension, the second is the PROFILE dimension. This is to allow K-matrix structures to be read in with the same function.
UNITS: N/A
TYPE: CRTM_Surface_type
DIMENSION: Rank-1 (M) or Rank-2 (L x M)
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages being printed to standard output (or the message log file if the Message_Log optional argument is used.) By default, INFORMATION messages are printed. If QUIET = 0, INFORMATION messages are OUTPUT. QUIET = 1, INFORMATION messages are SUPPRESSED.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL
Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Channels: The number of channels for which data was read. Note that this value will always be 0 for a profile-only dataset-- it only has meaning for K-matrix data.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

n_Profiles: The number of profiles for which data was read.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the version control Id
 field for the module.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
 The error codes are defined in the Message_Handler module.
 If == SUCCESS the Binary file read was successful
 == FAILURE an unrecoverable error occurred.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Surface argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.10 CRTM_Write_Surface_Binary interface

NAME:

CRTM_Write_Surface_Binary

PURPOSE:

Function to write Binary format Surface files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Write_Surface_Binary( Filename, &
                                           Surface, &
                                           Quiet =Quiet, &
                                           RCS_Id =RCS_Id, &
                                           Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an output
 Surface format data file.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

Surface: Structure containing the Surface data to write.
 Note the following meanings attributed to the dimensions of the structure array:
 Rank-1: M profiles.
 Only profile data are to be read in. The file does not contain channel information. The dimension of the structure is understood to be the PROFILE dimension.
 Rank-2: L channels x M profiles
 Channel and profile data are to be read in. The file contains both channel and profile information. The first dimension of the structure is the CHANNEL dimension, the second is the PROFILE dimension. This is to allow K-matrix structures to be read in with the same function.
 UNITS: N/A
 TYPE: CRTM_Surface_type
 DIMENSION: Rank-1 (M) or Rank-2 (L x M)
 ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages being printed to standard output (or the message log file if the Message_Log optional argument is used.) By default, INFORMATION messages are printed.
 If QUIET = 0, INFORMATION messages are OUTPUT.
 QUIET = 1, INFORMATION messages are SUPPRESSED.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the version control Id field for the module.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the Binary file write was successful

== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

- If the output file already exists, it is overwritten.
- If an error occurs *during* the write phase, the output file is deleted before returning to the calling routine.

DRAFT

A.5.11 SensorData Structure

```

TYPE :: CRTM_SensorData_type
  INTEGER :: n_Allocates = 0
  ! Dimension values
  INTEGER :: n_Channels = 0 ! L
  ! The WMO sensor ID of the sensor for which the data is to be used
  INTEGER :: Select_WMO_Sensor_Id = INVALID_WMO_SENSOR_ID
  ! The data sensor IDs and channels
  CHARACTER(STRLEN), POINTER :: Sensor_Id(:)      => NULL() ! L
  INTEGER,                POINTER :: WMO_Satellite_ID(:) => NULL() ! L
  INTEGER,                POINTER :: WMO_Sensor_ID(:)   => NULL() ! L
  INTEGER,                POINTER :: Sensor_Channel(:)  => NULL() ! L
  ! The sensor brightness temperatures
  REAL(fp),               POINTER :: Tb(:) => NULL() ! L
END TYPE CRTM_SensorData_type

```

Figure A.6: CRTM_SensorData_type structure definition.

Component	Description	Units	Dimensions
n_Channels	Number of channels to use in SfcOptics emissivity algorithms (L)	N/A	Scalar
Select_WMO_Sensor_Id	The WMO Sensor Id value of the sensor for which the data is to be used.	N/A	Scalar
Sensor_Id	The sensor id character string for each channel of data	N/A	L
WMO_Satellite_Id	The WMO satellite Id for each channel of data	N/A	L
WMO_Sensor_Id	The WMO sensor Id for each channel of data	N/A	L
Sensor_Channel	The channel number for each channel of data	N/A	L
Tb	The brightness temperature measurements for each channel	Kelvin	L

Table A.12: CRTM SensorData structure component description.

A.5.12 CRTM_Associated_SensorData interface

NAME:

CRTM_Associated_SensorData

PURPOSE:

Function to test the association status of the pointer members of a CRTM_SensorData structure.

CALLING SEQUENCE:

```
Association_Status = CRTM_Associated_SensorData( SensorData      , &
                                                ANY_Test=Any_Test  )
```

INPUT ARGUMENTS:

SensorData: SensorData structure which is to have its pointer member's association status tested.
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the SensorData structure pointer members are associated. The default is to test if ALL the pointer members are associated.
If ANY_Test = 0, test if ALL the pointer members are associated. (DEFAULT)
ANY_Test = 1, test if ANY of the pointer members are associated.

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the SensorData pointer members.
.TRUE. - if ALL the SensorData pointer members are associated, or if the ANY_Test argument is set and ANY of the SensorData pointer members are associated.
.FALSE. - some or all of the SensorData pointer members are NOT associated.
UNITS: N/A
TYPE: LOGICAL
DIMENSION: Scalar

A.5.13 CRTM_Allocate_SensorData interface

NAME:

CRTM_Allocate_SensorData

PURPOSE:

Function to allocate the pointer members of a CRTM SensorData data structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_SensorData( n_Channels      , &
                                         SensorData      , &
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Channels: The number of channels in the SensorData structure.
Must be > 0.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

SensorData: SensorData structure with allocated pointer members
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
Messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output Messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the structure pointer allocations were
successful
== FAILURE - an error occurred, or
- the structure internal allocation counter
is not equal to one (1) upon exiting this
function. This value is incremented and
decremented for every structure allocation
and deallocation respectively.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output SensorData argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.14 CRTM_Destroy_SensorData interface

NAME:

CRTM_Destroy_SensorData

PURPOSE:

Function to re-initialize the scalar and pointer members of SensorData data structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy_SensorData( SensorData      , &
                                         Message_Log=Message_Log )
```

OUTPUT ARGUMENTS:

SensorData: Re-initialized SensorData structure.
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to zero (0) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output SensorData argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.15 CRTM_Assign_SensorData interface

NAME:

CRTM_Assign_SensorData

PURPOSE:

Function to copy valid SensorData structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_SensorData( SensorData_in      , &
                                         SensorData_out     , &
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

SensorData_in: SensorData structure which is to be copied.
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Scalar OR Rank-1
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

SensorData_out: Copy of the input structure, SensorData_in.
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Same as SensorData_in
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output SensorData argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.5.16 CRTM_Equal_SensorData interface

NAME:

CRTM_Equal_SensorData

PURPOSE:

Function to test if two SensorData structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_SensorData( SensorData_LHS      , &
                                       SensorData_RHS      , &
                                       ULP_Scale   =ULP_Scale , &
                                       Check_All   =Check_All  , &
                                       Message_Log=Message_Log )
```

INPUT ARGUMENTS:

SensorData_LHS: SensorData structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.
IF (SensorData_LHS == SensorData_RHS).
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

SensorData_RHS: SensorData structure to be compared to; equivalent to right-hand side of a lexical comparison, e.g.
IF (SensorData_LHS == SensorData_RHS).
UNITS: N/A
TYPE: CRTM_SensorData_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Check_All: Set this argument to check ALL the floating point channel data of the SensorData structures. The default action is return with a FAILURE status as soon as

any difference is found. This optional argument can be used to get a listing of ALL the differences between data in SensorData structures.

If == 0, Return with FAILURE status as soon as ANY difference is found *DEFAULT*

== 1, Set FAILURE status if ANY difference is found, but continue to check ALL data.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: None

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structures were equal
== FAILURE - an error occurred, or
- the structures were different.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

A.5.17 CRTM_RCS_Id_SensorData interface

NAME:

CRTM_RCS_Id_SensorData

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_SensorData(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.

UNITS: N/A

TYPE: CHARACTER(*)

DIMENSION: Scalar

ATTRIBUTES: INTENT(OUT)

DRAFT

A.6 GeometryInfo Structure

```
TYPE :: CRTM_GeometryInfo_type
! User Input
! -----
! Earth location
REAL(fp) :: Longitude      = ZERO
REAL(fp) :: Latitude       = ZERO
REAL(fp) :: Surface_Altitude = ZERO
! Field of view index (1-nFOV)
INTEGER  :: iFOV = 0
! Sensor angle information
REAL(fp) :: Sensor_Scan_Angle    = ZERO
REAL(fp) :: Sensor_Zenith_Angle  = ZERO
REAL(fp) :: Sensor_Azimuth_Angle = ZERO
! Source angle information
REAL(fp) :: Source_Zenith_Angle  = 100.0_fp ! Below horizon
REAL(fp) :: Source_Azimuth_Angle = ZERO
! Flux angle information
REAL(fp) :: Flux_Zenith_Angle = DIFFUSIVITY_ANGLE
! Derived from User Input
! -----
! Default distance ratio
REAL(fp) :: Distance_Ratio = EARTH_RADIUS/(EARTH_RADIUS + SATELLITE_HEIGHT)
! Sensor angle information
REAL(fp) :: Sensor_Scan_Radian    = ZERO
REAL(fp) :: Sensor_Zenith_Radian  = ZERO
REAL(fp) :: Sensor_Azimuth_Radian = ZERO
REAL(fp) :: Secant_Sensor_Zenith  = ZERO
! Source angle information
REAL(fp) :: Source_Zenith_Radian  = ZERO
REAL(fp) :: Source_Azimuth_Radian = ZERO
REAL(fp) :: Secant_Source_Zenith  = ZERO
! Flux angle information
REAL(fp) :: Flux_Zenith_Radian = DIFFUSIVITY_RADIAN
REAL(fp) :: Secant_Flux_Zenith = SECANT_DIFFUSIVITY
END TYPE CRTM_GeometryInfo_type
```

Figure A.7: CRTM_GeometryInfo_type structure definition.

Component	Description	Units	Dimensions
Longitude	Earth longitude	deg. E (0→360)	Scalar
Latitude	Earth latitude	deg. N (-90→+90)	Scalar
Surface_Altitude	Altitude of the Earth's surface at the specified lon/lat location	metres (m)	Scalar
iFOV	The scan line FOV index	N/A	Scalar
Sensor_Scan_Angle	The sensor scan angle from nadir. See fig.A.8	degrees	Scalar
Sensor_Zenith_Angle	The sensor zenith angle of the FOV. See fig.A.9	degrees	Scalar
Sensor_Azimuth_Angle	The sensor azimuth angle is the angle subtended by the horizontal projection of a direct line from the satellite to the FOV and the North-South axis measured clockwise from North. See fig.A.10	deg. from N	Scalar
Source_Zenith_Angle	The source zenith angle. The source is typically the Sun (IR/VIS) or Moon (MW/VIS) [only solar source valid in current release] See fig.A.11	degrees	Scalar
Source_Azimuth_Angle	The source azimuth angle is the angle subtended by the horizontal projection of a direct line from the source to the FOV and the North-South axis measured clockwise from North. See fig.A.12	deg. from N	Scalar
Flux_Zenith_Angle	The zenith angle used to approximate downwelling flux transmissivity. If not set, the default value is that of the diffusivity approximation, such that $\sec(F) = 5/3$. Maximum allowed value is determined from $\sec(F) = 9/4$	degrees	Scalar

Table A.13: Description of CRTM GeometryInfo structure components defined by the user.

Component	Description	Units	Dimensions
Distance_Ratio	<p>The ratio of the radius of the earth at the FOV location to the sum of the radius of the earth at nadir, and the satellite altitude;</p> $r = \frac{R_e(FOV)}{R_e(nadir) + h}$ <p>Note that this quantity is actually computed using the user input sensor scan and zenith angles;</p> $r = \frac{\sin(\theta_{scan})}{\sin(\theta_{zenith})}$	N/A	Scalar
Sensor_Scan_Radian	The sensor scan angle in radians	radians	Scalar
Sensor_Zenith_Radian	The sensor scan angle in radians	radians	Scalar
Sensor_Azimuth_Radian	The sensor azimuth angle in radians	radians	Scalar
Secant_Sensor_Zenith	The secant of the sensor zenith angle	N/A	Scalar
Source_Zenith_Radian	The source zenith angle in radians	radians	Scalar
Source_Azimuth_Radian	The source azimuth angle in radians	radians	Scalar
Secant_Source_Zenith	The secant of the source zenith angle	N/A	Scalar
Flux_Zenith_Radian	The flux zenith angle in radians	radians	Scalar
Secant_Flux_Zenith	The secant of the flux zenith angle	N/A	Scalar

Table A.14: Description of CRTM GeometryInfo structure components derived from user inputs.

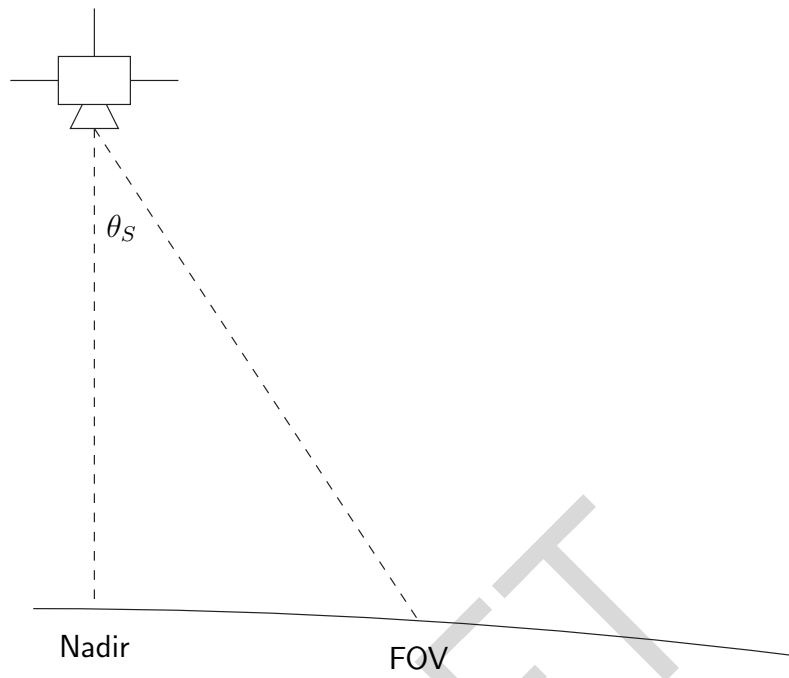


Figure A.8: Definition of GeometryInfo sensor scan angle component.

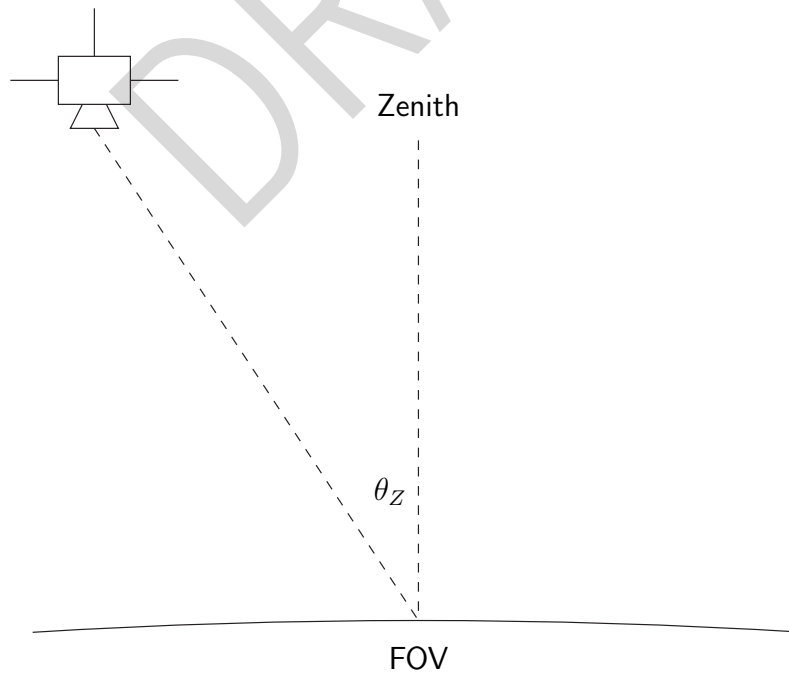


Figure A.9: Definition of GeometryInfo sensor zenith angle component.

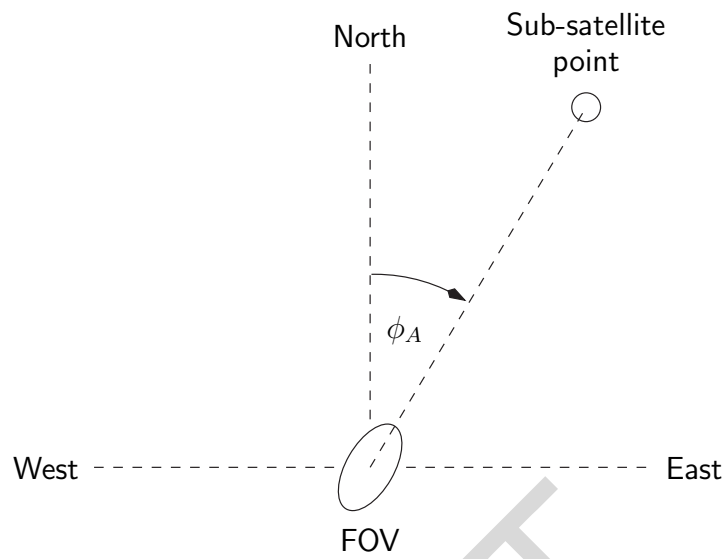


Figure A.10: Definition of GeometryInfo sensor azimuth angle component.

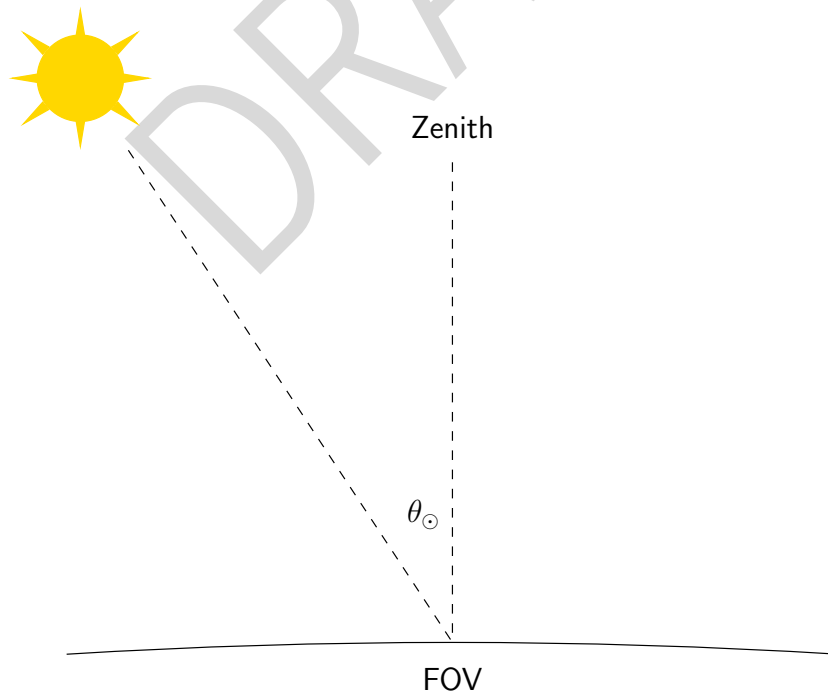


Figure A.11: Definition of GeometryInfo source zenith angle component.

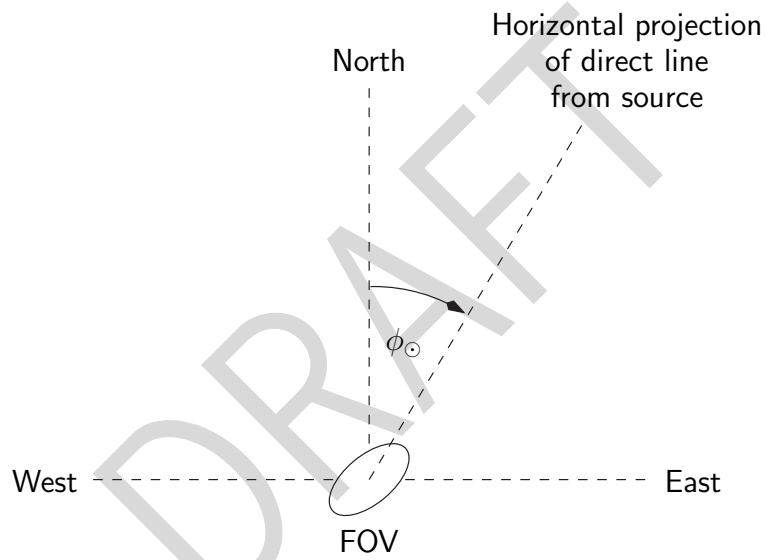


Figure A.12: Definition of GeometryInfo source azimuth angle component.

A.6.1 CRTM_Compute_GeometryInfo interface

NAME:

CRTM_Compute_GeometryInfo

PURPOSE:

Function to compute the derived geometry from the user specified components of the CRTM GeometryInfo structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Compute_GeometryInfo( GeometryInfo      , &
                                           Message_Log=Message_Log )
```

INPUT ARGUMENTS:

GeometryInfo: The GeometryInfo structure containing the user defined inputs, in particular the angles.
UNITS: N/A
TYPE: CRTM_GeometryInfo_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN OUT)

OUTPUT ARGUMENTS:

GeometryInfo: The GeometryInfo structure with the derived angle components filled..
UNITS: N/A
TYPE: CRTM_GeometryInfo_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to the screen.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the ERROR_HANDLER module.
If == SUCCESS the computation was successful
== WARNING invalid data was found, but altered to default.
== FAILURE invalid data was found
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

This function changes the values of the derived components of the GeometryInfo structure argument.

A.6.2 CRTM_RCS_ID_GeometryInfo interface

NAME:

CRTM_RCS_ID_GeometryInfo

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_GeometryInfo(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id:	Character string containing the Revision Control System Id field for the module.
UNITS:	N/A
TYPE:	CHARACTER(*)
DIMENSION:	Scalar
ATTRIBUTES:	INTENT(OUT)

A.7 RTSolution Structure

```
TYPE :: CRTM_RTSolution_type
  INTEGER :: n_Allocates = 0
  ! Dimensions
  INTEGER :: n_Layers = 0 ! K
  ! Internal variables. Users do not need to worry about these.
  LOGICAL :: Scattering_Flag = .TRUE.
  INTEGER :: n_Full_Streams = 0
  INTEGER :: n_Stokes = 0
  ! Forward radiative transfer intermediate results for a single channel
  !   These components are not defined when they are used as TL, AD
  !   and K variables
  REAL(fp) :: Surface_Emissivity = ZERO
  REAL(fp) :: Up_Radiance = ZERO
  REAL(fp) :: Down_Radiance = ZERO
  REAL(fp) :: Down_Solar_Radiance = ZERO
  REAL(fp) :: Surface_Planck_Radiance = ZERO
  REAL(fp), POINTER :: Upwelling_Radiance(:) => NULL() ! K
  ! The layer optical depths
  REAL(fp), POINTER :: Layer_Optical_Depth(:) => NULL() ! K
  ! Radiative transfer results for a single channel/node
  REAL(fp) :: Radiance = ZERO
  REAL(fp) :: Brightness_Temperature = ZERO
END TYPE CRTM_RTSolution_type
```

Figure A.13: CRTM_RTSolution_type structure definition.

Component	Description	Units	Dimensions
n_Layers	Number of atmospheric profile layers (K)	N/A	Scalar
Surface_Emissivity	The computed surface emissivity	N/A	Scalar
Up_Radiance	The atmospheric portion of the upwelling radiance	mW/(m ² .sr.cm ⁻¹)	Scalar
Down_Radiance	The atmospheric portion of the downwelling radiance	mW/(m ² .sr.cm ⁻¹)	Scalar
Down_Solar_Radiance	The downwelling direct solar radiance	mW/(m ² .sr.cm ⁻¹)	Scalar
Surface_Planck_Radiance	The surface radiance	mW/(m ² .sr.cm ⁻¹)	Scalar
Upwelling_Radiance	The upwelling radiance profile, including the reflected downwelling and surface contributions.	mW/(m ² .sr.cm ⁻¹)	K
Layer_Optical_Depth	The layer optical depth profile	N/A	K
Radiance	The sensor radiance	mW/(m ² .sr.cm ⁻¹)	Scalar
Brightness_Temperature	The sensor brightness temperature	Kelvin	Scalar

Table A.15: CRTM RTSolution structure component description

A.7.1 CRTM_Associated_RTSolution interface

NAME:

CRTM_Associated_RTSolution

PURPOSE:

Function to test the association status of the pointer members of a CRTM RTSolution structure.

CALLING SEQUENCE:

```
Association_Status = CRTM_Associated_RTSolution( RTSolution      , &
                                                ANY_Test=Any_Test )
```

INPUT ARGUMENTS:

RTSolution: RTSolution structure which is to have its pointer member's association status tested.
UNITS: N/A
TYPE: CRTM_RTSolution_type
DIMENSION: Scalar, Rank-1, OR Rank-2 array
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the RTSolution structure pointer members are associated. The default is to test if ALL the pointer members are associated.
 If ANY_Test = 0, test if ALL the pointer members are associated. (DEFAULT)
 ANY_Test = 1, test if ANY of the pointer members are associated.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the RTSolution pointer members.
 .TRUE. - if ALL the RTSolution pointer members are associated, or if the ANY_Test argument is set and ANY of the RTSolution pointer members are associated.
 .FALSE. - some or all of the RTSolution pointer members are NOT associated.
UNITS: N/A
TYPE: LOGICAL
DIMENSION: Same as input RTSolution argument

A.7.2 CRTM_Allocate_RTSolution interface

NAME:

CRTM_Allocate_RTSolution

PURPOSE:

Function to allocate the pointer members of the CRTM RTSolution data structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_RTSolution( n_Layers           , &
                                         RTSolution          , &
                                         Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Layers: Number of atmospheric layers
 Must be > 0
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

RTSolution: RTSolution structure with allocated pointer members.
 Upon allocation, all pointer members are initialized to
 a value of zero.

The following chart shows the allowable dimension combinations for the calling routine, where

L == number of channels
 M == number of profiles

Input	Output
n_Layers	RTSolution
dimension	dimension

scalar	scalar
scalar	Rank-1 (L or M)
scalar	Rank-2 (L x M)

These multiple interfaces are supplied purely for ease of use depending on how it's used.

UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Scalar, Rank-1, or Rank-2
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
 messages will be logged. If not specified, or if an
 error occurs opening the log file, the default action
 is to output messages to standard output.
 UNITS: N/A

TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
 The error codes are defined in the Message_Handler module.
 If == SUCCESS the structure re-initialisation was successful
 == FAILURE - an error occurred, or
 - the structure internal allocation counter
 is not equal to one (1) upon exiting this
 function. This value is incremented and
 decremented for every structure allocation
 and deallocation respectively.

UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output RTSolution argument is IN OUT rather than
 just OUT. This is necessary because the argument may be defined upon
 input. To prevent memory leaks, the IN OUT INTENT is a must.

A.7.3 CRTM_Destroy_RTSolution interface

NAME:

CRTM_Destroy_RTSolution

PURPOSE:

Function to re-initialize the scalar and pointer members of a CRTM
 RTSolution data structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Destroy_RTSolution( RTSolution      , &
                                       Message_Log=Message_Log )
```

OUTPUT ARGUMENTS:

RTSolution: Re-initialized RTSolution structure.
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Scalar, Rank-1, or Rank-2
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
 Messages will be logged. If not specified, or if an
 error occurs opening the log file, the default action
 is to output Messages to standard output.
 UNITS: N/A
 TYPE: CHARACTER(*)

DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter
is not equal to zero (0) upon exiting this
function. This value is incremented and
decremented for every structure allocation
and deallocation respectively.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output RTSolution argument is IN OUT rather than
just OUT. This is necessary because the argument may be defined upon
input. To prevent memory leaks, the IN OUT INTENT is a must.

A.7.4 CRTM_Assign_RTSolution interface

NAME:

CRTM_Assign_RTSolution

PURPOSE:

Function to copy valid CRTM RTSolution structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_RTSolution( RTSolution_in      , &  
                                       RTSolution_out      , &  
                                       Message_Log=Message_Log )
```

INPUT ARGUMENTS:

RTSolution_in: RTSolution structure which is to be copied.
UNITS: N/A
TYPE: CRTM_RTSolution_type
DIMENSION: Scalar, Rank-1, or Rank-2
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

RTSolution_out: Copy of the input structure, RTSolution_in.
UNITS: N/A
TYPE: CRTM_RTSolution_type
DIMENSION: Same as input RTSolution_in
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output RTSolution argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.7.5 CRTM_Equal_RTSolution interface

NAME:

CRTM_Equal_RTSolution

PURPOSE:

Function to test if two RTSolution structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_RTSolution( RTSolution_LHS      ,  
                                      RTSolution_RHS      ,  
                                      ULP_Scale            =ULP_Scale      ,  
                                      Percent_Difference=Percent_Difference,  
                                      Check_All             =Check_All       ,  
                                      Check_Intermediate=Check_Intermediate,  
                                      Message_Log           =Message_Log
```

INPUT ARGUMENTS:

RTSolution_LHS: RTSolution structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.
IF (RTSolution_LHS == RTSolution_RHS).
UNITS: N/A
TYPE: CRTM_RTSolution_type
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

RTSolution_RHS: RTSolution structure to be compared to; equivalent to right-hand side of a lexical comparison, e.g.
 IF (RTSolution_LHS == RTSolution_RHS).
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default value is 1.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Percent_Difference: Percentage difference value to use in comparing the numbers rather than testing within some numerical limit. The ULP_Scale argument is ignored if this argument is specified.
 UNITS: N/A
 TYPE: REAL(fp)
 DIMENSION: Scalar
 ATTRIBUTES: OPTIONAL, INTENT(IN)

Check_All: Set this argument to check ALL the floating point channel data of the RTSolution structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in RTSolution structures.
 If == 0, Return with FAILURE status as soon as ANY difference is found *DEFAULT*
 == 1, Set FAILURE status if ANY difference is found, but continue to check ALL data.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Check_Intermediate: Set this argument to check the intermediate results held in the RTSolution structure. The default action does NOT check these components.
 If == 0, Intermediate result components not checked for equality. *DEFAULT*
 == 1, Intermediate result components ARE checked for equality. Note that this could generate a lot of comparison failures.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.

UNITS: None
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structures were equal
== FAILURE - an error occurred, or
- the structures were different.

UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

A.7.6 CRTM_RCS_ID_RTSolution interface

NAME:

CRTM_RCS_ID_RTSolution

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_RTSolution(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id: Character string containing the Revision Control System Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(OUT)

A.7.7 CRTM_Inquire_RTSolution_Binary interface

NAME:

CRTM_Inquire_RTSolution_Binary

PURPOSE:

Function to inquire Binary format CRTM RTSolution structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Inquire_RTSolution_Binary( Filename           , &
                                              n_Channels =n_Channels , &
                                              n_Profiles =n_Profiles , &
                                              RCS_Id      =RCS_Id      , &
                                              Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an
RTSolution format data file to read.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Channels: The number of spectral channels for which there is
data in the file. Note that this value will always
be 0 for a profile-only RTSolution dataset-- it only
has meaning for K-matrix RTSolution data.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

n_Profiles: The number of atmospheric profiles in the data file.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the version control Id
field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.

The error codes are defined in the Message_Handler module.
 If == SUCCESS the Binary file inquire was successful
 == FAILURE an unrecoverable error occurred.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.7.8 CRTM_Read_RTSolution_Binary interface

NAME:

CRTM_Read_RTSolution_Binary

PURPOSE:

Function to read Binary format CRTM RTSolution structure files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Read_RTSolution_Binary( Filename           , &
                                           RTSolution          , &
                                           Quiet               , &
                                           n_Channels          , &
                                           n_Profiles          , &
                                           RCS_Id              , &
                                           Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an
 RTSolution format data file to read.
 UNITS: N/A
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

RTSolution: Structure array containing the RTSolution data. Note
 the rank is CHANNELS x PROFILES.
 UNITS: N/A
 TYPE: CRTM_RTSolution_type
 DIMENSION: Rank-2 (L x M)
 ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
 being printed to standard output (or the message
 log file if the Message_Log optional argument is
 used.) By default, INFORMATION messages are printed.
 If QUIET = 0, INFORMATION messages are OUTPUT.
 QUIET = 1, INFORMATION messages are SUPPRESSED.
 UNITS: N/A
 TYPE: INTEGER

DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

n_Channels: The number of channels for which data was read.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

n_Profiles: The number of profiles for which data was read.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

RCS_Id: Character string containing the version control Id field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
If == SUCCESS the Binary file read was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output RTSolution argument is IN OUT rather than just OUT. This is necessary because the argument may be defined on input. To prevent memory leaks, the IN OUT INTENT is a must.

A.7.9 CRTM_Write_RTSolution_Binary interface

NAME:

CRTM_Write_RTSolution_Binary

PURPOSE:

Function to write Binary format RTSolution files.

CALLING SEQUENCE:

```
Error_Status = CRTM_Write_RTSolution_Binary( Filename           , &
                                             RTSolution         , &
                                             Quiet              , &
                                             RCS_Id             , &
                                             Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Filename: Character string specifying the name of an output
RTSolution format data file.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

RTSolution: Structure containing the RTSolution data to write.
Note the rank is CHANNELS x PROFILES.
UNITS: N/A
TYPE: CRTM_RTSolution_type
DIMENSION: Rank-2 (L x M)
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

Quiet: Set this argument to suppress INFORMATION messages
being printed to standard output (or the message
log file if the Message_Log optional argument is
used.) By default, INFORMATION messages are printed.
If QUIET = 0, INFORMATION messages are OUTPUT.
QUIET = 1, INFORMATION messages are SUPPRESSED.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any
messages will be logged. If not specified, or if an
error occurs opening the log file, the default action
is to output messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

OPTIONAL OUTPUT ARGUMENTS:

RCS_Id: Character string containing the version control Id
field for the module.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: OPTIONAL, INTENT(OUT)

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status.
The error codes are defined in the Message_Handler module.
If == SUCCESS the Binary file write was successful
== FAILURE an unrecoverable error occurred.
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar

SIDE EFFECTS:

- If the output file already exists, it is overwritten.
- If an error occurs *during* the write phase, the output file is deleted before returning to the calling routine.

DRAFT

A.8 Options Structure

```

TYPE :: CRTM_Options_type
  INTEGER :: n_Allocates = 0
  ! Dimensions
  INTEGER :: n_Channels = 0 ! L dimension
  ! Index into channel-specific components
  INTEGER :: Channel = 0
  ! Emissivity optional arguments
  INTEGER      :: Emissivity_Switch = NOT_SET
  REAL(fp), POINTER :: Emissivity(:)  => NULL() ! L
  ! Direct reflectivity optional arguments
  INTEGER      :: Direct_Reflectivity_Switch = NOT_SET
  REAL(fp), POINTER :: Direct_Reflectivity(:)  => NULL() ! L
  ! Antenna correction application
  INTEGER :: Antenna_Correction = NOT_SET
END TYPE CRTM_Options_type

```

Figure A.14: CRTM_Options_type structure definition.

Component	Description	Units	Dimensions
n_Channels	Number of sensor channels (L).	N/A	Scalar
Channel	Index into channel-specific components.	N/A	Scalar
Emissivity_Switch	Switch to apply user-defined surface emissivity. Valid values: NOT_SET: Calculate emissivity (default). SET: Use user-defined emissivity	N/A	Scalar
Emissivity	User-defined surface emissivity for each sensor channel.	N/A	L
Direct_Reflectivity_Switch	Switch to apply user-defined reflectivity for downwelling source (e.g. solar). This switch is ignored unless the Emissivity_Switch is also set. Valid values: NOT_SET: Calculate reflectivity (default). SET: Use user-defined reflectivity	N/A	Scalar
Direct_Reflectivity	User-defined direct reflectivity for downwelling source for each sensor channel.	N/A	L
Antenna_Correction	Switch to apply antenna correction for select microwave instruments.	N/A	Scalar

Table A.16: CRTM Options structure component description

A.8.1 CRTM_Associated_Options interface

NAME:

CRTM_Associated_Options

PURPOSE:

Function to test the association status of the pointer members of a CRTM Options structure.

CALLING SEQUENCE:

```
Association_Status = CRTM_Associated_Options( Options           , &
                                             ANY_Test=Any_Test  )
```

INPUT ARGUMENTS:

Options: Options structure which is to have its pointer member's association status tested.

UNITS: N/A

TYPE: CRTM_Options_type

DIMENSION: Scalar or Rank-1 array

ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ANY_Test: Set this argument to test if ANY of the Options structure pointer members are associated. The default is to test if ALL the pointer members are associated.

If ANY_Test = 0, test if ALL the pointer members are associated. (DEFAULT)

ANY_Test = 1, test if ANY of the pointer members are associated.

UNITS: N/A

TYPE: INTEGER

DIMENSION: Scalar

ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Association_Status: The return value is a logical value indicating the association status of the Options pointer members.

.TRUE. - if ALL the Options pointer members are associated, or if the ANY_Test argument is set and ANY of the Options pointer members are associated.

.FALSE. - some or all of the Options pointer members are NOT associated.

UNITS: N/A

TYPE: LOGICAL

DIMENSION: Same as input Options argument

A.8.2 CRTM_Allocate_Options interface

NAME:

CRTM_Allocate_Options

PURPOSE:

Function to allocate the pointer members of a CRTM Options data structure.

CALLING SEQUENCE:

```
Error_Status = CRTM_Allocate_Options( n_Channels      , &
                                       Options          , &
                                       Message_Log=Message_Log )
```

INPUT ARGUMENTS:

n_Channels: Number of sensor channels
Must be > 0
UNITS: N/A
TYPE: INTEGER
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Options: Options structure with allocated pointer members.
Upon allocation, all pointer members are initialized to a value of zero.
UNITS: N/A
TYPE: CRTM_Options_type
DIMENSION: Scalar or Rank-1
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any Messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output Messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
If == SUCCESS the structure pointer allocations were successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to one (1) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.
UNITS: N/A

TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Options argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.8.3 CRTM_Destroy_Options interface

NAME:

CRTM_Destroy_Options

PURPOSE:

Function to re-initialize the scalar and pointer members of a CRTM Options data structure.

CALLING SEQUENCE:

Error_Status = CRTM_Destroy_Options(Options , &
Message_Log=Message_Log)

OUTPUT ARGUMENTS:

Options: Re-initialized Options structure.
UNITS: N/A
TYPE: CRTM_Options_type
DIMENSION: Scalar or Rank-1
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any Messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output Messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure re-initialisation was successful
== FAILURE - an error occurred, or
- the structure internal allocation counter is not equal to zero (0) upon exiting this function. This value is incremented and decremented for every structure allocation and deallocation respectively.
UNITS: N/A
TYPE: INTEGER

DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Options argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.8.4 CRTM_Assign_Options interface

NAME:

CRTM_Assign_Options

PURPOSE:

Function to copy valid CRTM Options structures.

CALLING SEQUENCE:

```
Error_Status = CRTM_Assign_Options( Options_in      , &
                                   Options_out      , &
                                   Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Options_in: Options structure which is to be copied.
UNITS: N/A
TYPE: CRTM_Options_type
DIMENSION: Scalar or Rank-1
ATTRIBUTES: INTENT(IN)

OUTPUT ARGUMENTS:

Options_out: Copy of the input structure, Options_in.
UNITS: N/A
TYPE: CRTM_Options_type
DIMENSION: Same as Options_in
ATTRIBUTES: INTENT(IN OUT)

OPTIONAL INPUT ARGUMENTS:

Message_Log: Character string specifying a filename in which any Messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output Messages to standard output.
UNITS: N/A
TYPE: CHARACTER(*)
DIMENSION: Scalar
ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module. If == SUCCESS the structure assignment was successful
== FAILURE an error occurred
UNITS: N/A

TYPE: INTEGER
DIMENSION: Scalar

COMMENTS:

Note the INTENT on the output Options argument is IN OUT rather than just OUT. This is necessary because the argument may be defined upon input. To prevent memory leaks, the IN OUT INTENT is a must.

A.8.5 CRTM_Equal_Options interface

NAME:

CRTM_Equal_Options

PURPOSE:

Function to test if two CRTM Options structures are equal.

CALLING SEQUENCE:

```
Error_Status = CRTM_Equal_Options( Options_LHS      , &  
                                   Options_RHS      , &  
                                   ULP_Scale   =ULP_Scale , &  
                                   Check_All   =Check_All  , &  
                                   Message_Log=Message_Log )
```

INPUT ARGUMENTS:

Options_LHS: Options structure to be compared; equivalent to the left-hand side of a lexical comparison, e.g.
 IF (Options_LHS == Options_RHS).
 In the context of the CRTM, rank-1 corresponds to an vector of profiles.
UNITS: N/A
TYPE: CRTM_Options_type
DIMENSION: Scalar or Rank-1 array
ATTRIBUTES: INTENT(IN)

Options_RHS: Options structure to be compared to; equivalent to right-hand side of a lexical comparison, e.g.
 IF (Options_LHS == Options_RHS).
UNITS: N/A
TYPE: CRTM_Options_type
DIMENSION: Same as Options_LHS
ATTRIBUTES: INTENT(IN)

OPTIONAL INPUT ARGUMENTS:

ULP_Scale: Unit of data precision used to scale the floating point comparison. ULP stands for "Unit in the Last Place," the smallest possible increment or decrement that can be made using a machine's floating point arithmetic. Value must be positive - if a negative value is supplied, the absolute value is used. If not specified, the default

value is 1.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Check_All: Set this argument to check ALL the floating point channel data of the Options structures. The default action is return with a FAILURE status as soon as any difference is found. This optional argument can be used to get a listing of ALL the differences between data in Options structures.
 If == 0, Return with FAILURE status as soon as ANY difference is found *DEFAULT*
 == 1, Set FAILURE status if ANY difference is found, but continue to check ALL data.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

Message_Log: Character string specifying a filename in which any messages will be logged. If not specified, or if an error occurs opening the log file, the default action is to output messages to standard output.
 UNITS: None
 TYPE: CHARACTER(*)
 DIMENSION: Scalar
 ATTRIBUTES: INTENT(IN), OPTIONAL

FUNCTION RESULT:

Error_Status: The return value is an integer defining the error status. The error codes are defined in the Message_Handler module.
 If == SUCCESS the structures were equal
 == FAILURE - an error occurred, or
 - the structures were different.
 UNITS: N/A
 TYPE: INTEGER
 DIMENSION: Scalar

A.8.6 CRTM_RCS_ID_Options interface

NAME:

CRTM_RCS_ID_Options

PURPOSE:

Subroutine to return the module RCS Id information.

CALLING SEQUENCE:

CALL CRTM_RCS_Id_Options(RCS_Id)

OUTPUT ARGUMENTS:

RCS_Id:	Character string containing the Revision Control System Id field for the module.
UNITS:	N/A
TYPE:	CHARACTER(*)
DIMENSION:	Scalar
ATTRIBUTES:	INTENT(OUT)

DRAFT

B

Valid Sensor Identifiers

DRAFT

B.1 Infrared instruments

DRAFT

Instrument	Sensor Id	Instrument	Sensor Id
Aqua AIRS (281ch. subset)	airs281.aqua	NOAA-16 HIRS/3	hirs3_n16
Aqua AIRS (324ch. subset)	airs324.aqua	NOAA-17 HIRS/3	hirs3_n17
Aqua AIRS Module-1a	airsM1a.aqua	NOAA-18 HIRS/4	hirs4_n18
Aqua AIRS Module-1b	airsM1b.aqua	MetOp-A HIRS/4	hirs4_metop-a
Aqua AIRS Module-2a	airsM2a.aqua	NOAA-19 HIRS/4	hirs4_n19
Aqua AIRS Module-2b	airsM2b.aqua	MetOp-A IASI (300ch. subset)	iasi300_metop-a
Aqua AIRS Module-3	airsM3.aqua	MetOp-A IASI (316ch. subset)	iasi316_metop-a
Aqua AIRS Module-4a	airsM4a.aqua	MetOp-A IASI (616ch. subset)	iasi616_metop-a
Aqua AIRS Module-4b	airsM4b.aqua	MetOp-A IASI Band 1	iasiB1_metop-a
Aqua AIRS Module-4c	airsM4c.aqua	MetOp-A IASI Band 2	iasiB2_metop-a
Aqua AIRS Module-4d	airsM4d.aqua	MetOp-A IASI Band 3	iasiB3_metop-a
Aqua AIRS Module-5	airsM5.aqua	MetOp-A IASI	iasi_metop-a
Aqua AIRS Module-6	airsM6.aqua	NPP CrIS Band 1	crisB1_npp
Aqua AIRS Module-7	airsM7.aqua	NPP CrIS Band 2	crisB2_npp
Aqua AIRS Module-8	airsM8.aqua	NPP CrIS Band 3	crisB3_npp
Aqua AIRS Module-9	airsM9.aqua	GOES-08 Imager	imgr_g08
Aqua AIRS Module-10	airsM10.aqua	GOES-09 Imager	imgr_g09
Aqua AIRS Module-11	airsM11.aqua	GOES-10 Imager	imgr_g10
Aqua AIRS Module-12	airsM12.aqua	GOES-11 Imager	imgr_g11
Aqua AIRS	airs.aqua	GOES-12 Imager	imgr_g12
TIROS-N AVHRR/2	avhrr2.tirosn	GOES-13 Imager	imgr_g13
NOAA-06 AVHRR/2	avhrr2_n06	GOES-R ABI	abi_gr
NOAA-07 AVHRR/2	avhrr2_n07	MTSAT-1R Imager	imgr_mt1r
NOAA-08 AVHRR/2	avhrr2_n08	Aqua MODIS	modis_aqua
NOAA-09 AVHRR/2	avhrr2_n09	Terra MODIS	modis.terra
NOAA-10 AVHRR/2	avhrr2_n10	MeteoSat-08 SEVIRI	seviri_m08
NOAA-11 AVHRR/2	avhrr2_n11	MeteoSat-09 SEVIRI	seviri_m09
NOAA-12 AVHRR/2	avhrr2_n12	MeteoSat-10 SEVIRI	seviri_m10
NOAA-14 AVHRR/2	avhrr2_n14	GOES-08 Sounder	sndr_g08
NOAA-15 AVHRR/3	avhrr3_n15	GOES-09 Sounder	sndr_g09
NOAA-16 AVHRR/3	avhrr3_n16	GOES-10 Sounder	sndr_g10
NOAA-17 AVHRR/3	avhrr3_n17	GOES-11 Sounder	sndr_g11
NOAA-18 AVHRR/3	avhrr3_n18	GOES-12 Sounder	sndr_g12
MetOp-A AVHRR/3	avhrr3_metop-a	GOES-13 Sounder	sndr_g13
NOAA-19 AVHRR/3	avhrr3_n19	TIROS-N SSU	ssu.tirosn
TIROS-N HIRS/2	hirs2.tirosn	NOAA-06 SSU	ssu_n06
NOAA-06 HIRS/2	hirs2_n06	NOAA-07 SSU	ssu_n07
NOAA-07 HIRS/2	hirs2_n07	NOAA-08 SSU	ssu_n08
NOAA-08 HIRS/2	hirs2_n08	NOAA-09 SSU	ssu_n09
NOAA-09 HIRS/2	hirs2_n09	NOAA-10 SSU	ssu_n11
NOAA-10 HIRS/2	hirs2_n10	NOAA-11 SSU	ssu_n14
NOAA-11 HIRS/2	hirs2_n11	GMS-5 VISSR (Detector A)	vissrDetA.gms5
NOAA-12 HIRS/2	hirs2_n12	Fengyun-3A VIRR	virr_fy3a
NOAA-14 HIRS/2	hirs2_n14	Fengyun-3A IRAS	iras_fy3a
NOAA-15 HIRS/3	hirs3_n15	Fengyun-3A MERSI	mersi_fy3a

Table B.1: CRTM sensor identifiers for infrared instruments.

B.2 Microwave instruments

Instrument	Sensor Id	Instrument	Sensor Id
Aqua AMSR-E	amsre_aqua	MetOp-A MHS	mhs_metop-a
Aqua AMSU-A	amsua_aqua	MetOp-B MHS	mhs_metop-b
Aqua HSB	hsb_aqua	MetOp-C MHS	mhs_metop-c
TIROS-N MSU	msu_tirosn	DMSP-08 SSM/I	ssmi_f08
NOAA-06 MSU	msu_n06	DMSP-10 SSM/I	ssmi_f10
NOAA-07 MSU	msu_n07	DMSP-11 SSM/I	ssmi_f11
NOAA-08 MSU	msu_n08	DMSP-13 SSM/I	ssmi_f13
NOAA-09 MSU	msu_n09	DMSP-14 SSM/I	ssmi_f14
NOAA-10 MSU	msu_n10	DMSP-15 SSM/I	ssmi_f15
NOAA-11 MSU	msu_n11	DMSP-13 SSM/T-1	ssmt1_f13
NOAA-12 MSU	msu_n12	DMSP-15 SSM/T-1	ssmt1_f15
NOAA-14 MSU	msu_n14	DMSP-14 SSM/T-2	ssmt2_f14
NOAA-15 AMSU-A	amsua_n15	DMSP-15 SSM/T-2	ssmt2_f15
NOAA-16 AMSU-A	amsua_n16	DMSP-16 SSMIS	ssmis_f16
NOAA-17 AMSU-A	amsua_n17	DMSP-17 SSMIS	ssmis_f17
NOAA-18 AMSU-A	amsua_n18	DMSP-18 SSMIS	ssmis_f18
NOAA-19 AMSU-A	amsua_n19	DMSP-19 SSMIS	ssmis_f19
MetOp-A AMSU-A	amsua_metop-a	DMSP-20 SSMIS	ssmis_f20
MetOp-B AMSU-A	amsua_metop-b	NPP ATMS	atms_npp
MetOp-C AMSU-A	amsua_metop-c	Coriolis WindSat	windsat_coriolis
NOAA-15 AMSU-B	amsub_n15	TRMM TMI	tmi_trmm
NOAA-16 AMSU-B	amsub_n16	GPM GMI	gmi_gpm
NOAA-17 AMSU-B	amsub_n17	Fengyun-3A MWRI	mwri_fy3a
NOAA-18 MHS	mhs_n18	Fengyun-3A MWHS	mwhs_fy3a
NOAA-19 MHS	mhs_n19	Fengyun-3A MWTS	mwts_fy3a

Table B.2: CRTM sensor identifiers for microwave instruments.

C

Utility Software Description

DRAFT

DRAFT

DRAFT

DRAFT