Generation of AUX_CAL Detailed Processing Model Input/Output data definition

A. Dabas, Météo-France

Table of content

Generation of AUX_CAL Detailed Processing Model Input/Output data definition 1
Table of content 3
1. Log 5
2. Reference documents 5
1. Acronyms 6
2. Introduction 6
3. Theoretical baseline. 7
3.1 Calibration functions C1 to C4. 7
3.2 Calibration coefficients KRay and KMie. 7
4. Detailed processing model 8
4.1 parameter Settings 8
4.2 Transmission curves of the double Fabry-Perot and the Fizeau 9
4.3 Computation of functions C1 to C4. 10
4.4 Computation of KRay and KMie. 12
4.5 Validity period 20
5. Input and output files 20
5.1 General product structure 20
5.2 Input files 21
5.3 Output file 23

1. Log

Date	Version	Author(s)	Comment
30/08/2009	0.1	A. Dabas	Initial version
27/07/2011	1.0	A. Dabas	- Parameter settings not hard-coded anymore but passed via the AUX_PAR_CL file - AUX_CAL data now output in AUX_CAL_L2 file. - Modification of equation (1) (does not depend on λ, and the collecting area of the telescope anymore.
17/01/2012	2.0	A. Dabas	Adaption to CM: - Modification of IRC_1A data reading. Frequency steps are now across 3 BRCs. - Modification of ISR_1A data reading. There are still 3 frequencies per RBC, but the number of measurements has changed, therefore the association of frequencies and measurements has changed. Update of Reference documents.
15/03/2012	2.1	A. Dabas	AE-IPF-179
05/04/2013	2.2	A. Dabas	– AE-IPF-190 – Correction of bug on convolution of spectra by transmission curves. – Screening of height-bins for estimation of K_{ray} and K_{mie}. – Update of references.
15/01/2014	2.3	A. Dabas	– Removal of calls to ISR and IRC 1A products – Possibility to input up to two AUX_MET products. – Introduction of a new threshold in AUX_PAR_CL (for Kray and Kmie estimations). – Update of references
22/07/2014	2.4	A.Dabas	– Update of DSDs in AUX_CAL file IODD
06/10/2014	2.5	A. Dabas	– Update of references. – Correction of equation (7).
21/01/2015	2.6	A. Dabas	– Update of acronyms – Size of T_min, T_max, P_min and P_max corrected (IPF201) – Check compatibility between IODD type and content of AUX_PAR_CL
10/03/2015	3.0	A. Dabas	– IPF 195 – IPF 198 (validity dates). – Update of references.
20/03/2015	3.0.1	A. Dabas	– Removal of call to ISR – Correction eq. 7
15/12/2015	3.2	A. Dabas	– IPF 217 – Add Goid/ellipsoid altitude offset to AUX_MET altitudes. – Add swidth to select molecular spectral model. – Update references.
30/06/2016	3.3	A. Dabas	– Geoid separation read from RRC rather than AUX_MET.
27/07/2016	3.3.1	A. Dabas	– Minor modification of reading of geoid separation. The middle one is taken instead of the second one.
10/07/2017	4.0	A. Dabas	– Update of references for the July 2017 delivery of the calibration suite. – Modification of Tenti spectrum: the shear viscosity is now tempreture dependant. – Tested with E2S 4.01 and L1B 7.00.

2. Reference documents

	Reference	Title	Version	Date
[CAL-DEF]	AE-TN-MFG-L2A-CAL	Calibration constants for the retrieval of optical properties of the atmosphere	1.0	27/03/2006
[L2BC-IODD]	AE-IF-ECMWF-L2BP-001	ADM-Aeolus Level-2B/2C Processor Input/Output Data Definitions (IODD) ICD	2.30	01/07/2016
[AUX-CSR]	AE-TN-MFG-L2P-CAL-003	Generation and update of AUX_CSR	4.0	24/07/2017
[L1B-IODD]	ADM-IC-52-1666	Aeolus Level 1b Processor and End-to-End Simulator: Input/Output Data Definitions Interface Control Document	4.07	02/05/2017
[AUX-RBC]	AE-TN-MFG-GS-0001	Generation of the RBC Auxiliary file: Detailed Processing Model	4.0	24/07/2017
[L1B-DPM]	ADM-MA-52-1800	L1B Processor Detailed Processing Model	3.07	21/07/2017
[L2A-ATBD]	AE-TN-IPSL-GS-001	ADM-Aeolus L2A Algorithm Theoretical Baseline Document	5.5	17/01/2017Ò

1. Acronyms

ACMF	Aeolus Calibration & Monitoring Facility
CSR	Coherent Spectral Registration
IRC	Instrument Response Calibration
ISR	Instrument Spectral Registration
MetPF	Meteorological Processing Facility
MRC	Mie Response Calibration
RRC	Rayleigh Response Calibration
WVM	Wind Velocity Measurement

2. Introduction

The present document describes the processing model of the AUX_CAL generator and details the input and output product files.

The AUX_CAL file shall be generated by the ACMF. It shall contain all the calibration functions and constants that characterize the energy balance of ADM. It is required by the L2A processor for cross-talk correction. It may also be used during the mission to monitor the performance of the lidar.

The functions and constants in the AUX_CAL file are defined by the following equations:

$$S_{Ray}(z) = K_{Ray}\left[C_1\big(P(z),T(z),\nu(z)\big)\beta_{mol}(z) + C_2\big(\nu(z)\big)\beta_{aer}(z)\right]\frac{T_{mol}^2(z)T_{aer}^2(z)}{R^2(z)}$$
$$S_{Mie}(z) = K_{Mie}\left[C_4\big(P(z),T(z),\nu(z)\big)\beta_{mol}(z) + C_3\big(\nu(z)\big)\beta_{aer}(z)\right]\frac{T_{mol}^2(z)T_{aer}^2(z)}{R^2(z)} \quad (1)$$

Here, $S_{Ray}(z)$ and $S_{Mie}(z)$ (in $[m^{-1}]$) are the normalized useful signals at the altitude z in $[m]$ in Rayleigh and Mie channels, $\beta_{mol}\,[m^{-1}sr^{-1}]$ and $\beta_{aer}\,[m^{-1}sr^{-1}]$ are the molecular and aerosol backscatter coefficients, $T_{mol}(z)$ and $T_{aer}(z)$ are the one-way atmospheric transmission between the satellite and the altitude z, $R(z)\,[m]$ is the satellite to target range, and $\nu(z)$ is the frequency of the light backscattered by the atmosphere.

The normalized useful signals are computed by the L1B processor. They are the signal strength detected on both Rayleigh and Mie channels divided by the total amount of emitted laser energy in $[mJ]$.

The functions C_1 and C_4 are the fraction of the molecular backscatter that is actually detected by the Rayleigh and Mie channels respectively. They depend on the pressure and temperature of the backscattering layer of the atmosphere and the frequency $\nu(z)$ through the shape and width of the molecular return spectrum. By convention, we set:

$$C_1(1000hPa, 300K, 0MHz) = C_4(1000hPa, 300K, 0MHz) = 1 \quad (2)$$

The functions C_2 and C_3 are the fraction of the aerosol backscatter that is actually detected in the Rayleigh and Mie channels respectively.

3. Theoretical baseline.

3.1 CALIBRATION FUNCTIONS C_1 TO C_4.

C_1, C_2, C_3, C_4 and K_{Ray} and K_{Mie} can be estimated as explained in [CAL-DEF]. C_1 and C_4 are computed by convolving the transmission characteristics of the Fabry-Perot and the Fizeau by the expected shape of the molecular return

$$C_1(P,T,\nu) = \frac{1}{K_1}\int \big(T_A(\mu) + T_B(\mu)\big) I_{mol}(P,T,\mu-\nu)d\mu$$
$$C_4(P,T,\nu) = \frac{1}{K_4}\int T_{Fiz}(\mu) I_{mol}(P,T,\mu-\nu)d\mu \quad (3)$$

Here, K_1 and K_4 are normalization constants. With the convention in equation (2):

$$K_1 = \int \big(T_A(\mu) + T_B(\mu)\big) I_{mol}(1000hPa, 300K, \mu)d\mu$$
$$K_4 = \int T_{Fiz}(\mu) I_{mol}(1000hPa, 300K, \mu)d\mu \quad (4)$$

As far as constants C_2 and C_3 are concerned, they are directly proportional to $T_A + T_B$ and T_{Fiz}:

$$C_2(\nu) = \frac{T_A(\nu) + T_B(\nu)}{K_1} \qquad and \qquad C_3(\nu) = \frac{T_{Fiz}(\nu)}{K_4} \quad (5)$$

Note that (5) assumes that the molecular spectrum $I_{mol}(\nu)$ has a unit area.

3.2 CALIBRATION COEFFICIENTS K_{Ray} AND K_{Mie}.

These coefficients are estimated from data acquired during an IRC. The idea is to compare the Mie and Rayleigh useful signals actually measured by the instrument in regions of the atmosphere where we can expect the aerosol backscatter is negligible (this condition can tested by looking at the scattering ratio estimated by the L1B processor from Mie signals), to predictions based on equation (1) and vertical profiles of atmospheric pressure and temperature provided by the L2_Met_12 generator and contained in AUX_MET_2B files (see section 5.1 in [L2BC-IODD]).

In case the aerosol backscatter and extinction are negligible, equation (1) simplifies to:

$$S_{Ray}(z) = K_{Ray}\hat{S}_{Ray}(z) \qquad S_{Mie}(z) = K_{Mie}\hat{S}_{Mie}(z)$$

where

$$\hat{S}_{Ray}(z) = C_1\big(P(z),T(z),\nu(z)\big)\beta_{mol}(z)\frac{T_{mol}^2(z)}{R^2(z)}$$
$$\hat{S}_{Ray}(z) = C_4\big(P(z),T(z),\nu(z)\big)\beta_{mol}(z)\frac{T_{mol}^2(z)}{R^2(z)} \quad (6)$$

The functions $\hat{S}_{Ray}(z)$ and $\hat{S}_{Mie}(z)$ can be "predicted" as:

- $R(z)$ is measured by the lidar and reported in the AUX_RRC_1B and AUX_MRC_1B files.
- The molecular backscatter β_{mol} and extinction coefficients α_{mol} are known functions of the atmospheric pressure and temperature (see equation 4.5 in [L2A-ATBD]):

$$\beta_{mol}(z)) = 1.38\left[\frac{550nm}{\lambda[nm]}\right]^{4.09}\left[\frac{P(z)[hPa]}{1013hPa}\frac{288K}{T(z)[K]}\right]10^{-6}m^{-1}sr^{-1} \quad (7)$$

$$\alpha(z) = \frac{8\pi}{3}\beta_{mol}(z)$$

- The transmission T_{mol} is equal to

$$T_{mol}(z) = \exp\left(-\int_{z}^{z_{sat}} \alpha_{mol}(x)dR(x)\right) \quad (8)$$

  where z_{sat} is the satellite altitude.

- The frequency $\nu(z)$ of the backscatter return is equal to the laser frequency as the lidar is pointing vertically in the IRC mode and vertical winds can be assumed negligible.

The coefficients K_{Ray} and K_{Mie} can thus be estimated by comparing the Rayeligh and Mie useful signals actually measured in one or several height bins to the "predictions" $\hat{S}_{Ray}(z)$ and $\hat{S}_{Mie}(z)$ integrated over the same height-bins with optical parameters α_{mol} and β_{mol} given by equation (7) with the pressure $P(z)$ and temperature $T(z)$ of the AUX_MET files. Denoting $S_{Ray}(k)$ and $S_{Mie}(k)$ the actual Rayleigh and Mie normalized useful signals measured in the k-th height-bin with bottom and top altitudes z_k and z_{k+1}, the estimation can be summarized by the equation:

$$\widehat{K}_{Ray} = \frac{S_{Ray}(k)}{\int_{z_k}^{z_{k+1}} \hat{S}_{Ray}(z)dR(z)} \quad \text{and} \quad \widehat{K}_{Mie} = \frac{S_{Mie}(k)}{\int_{z_k}^{z_{k+1}} \hat{S}_{Mie}(z)dR(z)} \quad (9)$$

In practice, this operation is done on all the valid height-bins of an IRC that are comprised in a pre-defined altitude interval $[z_{min}, z_{max}]$. The potential contamination of the height-bins by aerosols is checked with scattering ratio estimates provided for Mie height-bins by the L1B processor. Height-bins with a scattering ratio greater than a predefined threshold (set in the parameters setting file AUX_PAR_CL) are discarded. The bottom altitude z_{min} must be high enough so that the probability of a significant contamination of the useful signals by Mie light is small.

In principle, equation (9) can be applied to all the valid height-bins of any measuring mode (WVM, ISR…). The IRC mode offers the advantage it is done on a regular basis (once a week) and the instrument is pointing at nadir (so the frequency $\nu(z)$ is known).

4. Detailed processing model

4.1 PARAMETER SETTINGS

The processor parameter settings are summarized in Table 1. They are passed to the processor via the AUX_PAR_CL xml file. Its content is described in section 5.2.1. Parameter settings are collected in the $param$ structure.

Category	Name	Unit	Comment
Instrument	USR	GHz	Useful Spectral Range.
	FSRFP	GHz	Free Spectral Range of the Fabry-Perot interferometers
	FSRFiz	GHz	Free Spectral Range of the Fizeau interferometer
	Df	MHz	Frequency resolution
Ref_Grid	Dz_Ref	m	The three parameters define the vertical grid used for computing lidar responses:
	Zref_Min	m	

	Zref_Max	m	Zref = Zref_Min:Dz_Ref:Zref_Max
Atm_Grid	Tcal_Min	K	The three parameters define the temperatures for which functions C_1 and C_4 must be computed: Tcal = Tcal_Min:Tcal_Stp:Tcal:Max
	Tcal_Max	K	
	Tcal_Stp	K	
	Pcal_Min	hPa	The three parameters define the pressures for which functions C_1 and C_4 must be computed: Pcal = Pcal_Min:Pcal_Stp:Pcal:Max
	Pcal_Max	hPa	
	Pcal_Stp	hPa	
Matchup	Time_Max	s	Max time difference between an IRC observation and a matched-up atmospheric profile.
	Range_Max	km	Max distance between an IRC observation and a matched-up atmospheric profile.
	RBC_Spec_Model		Switch selecting the model used for the simulation molecular spectra. It can be either GAUSS (pure Rayleigh) or TENTI (Rayleigh-Brillouin).
Thresholds	Min_Freq_Steps_Valid		Minimum number of valid frequency steps required for the computation of AUX_CAL. If the number is not reached, the processor returns an error.
	Z_Min	m	Min altitude of height-bins retained for the computation of K_{Ray} and K_{Mie}.
	Z_Max	m	Max altitude of height-bins retained for the computation of K_{Ray} and K_{Mie}.
	Min_Signal_Level		Minimum normalized useful signal required for the computation of K_{Ray} and K_{Mie}.
	Max_Mie_Scat		Maximum value of the scattering ratio allowed for a height bin to be included in the computation of K_{Ray} and K_{Mie}.

Table 1: List of input parameters and parameter settings of the AUX_CAL generator.

4.2 TRANSMISSION CURVES OF THE DOUBLE FABRY-PEROT AND THE FIZEAU

After the parameter settings are declared, the processor reads the content of the currently valid AUX_CSR file. The AUX_CSR file is searched in the AUX directory of the calibration suite. If no AUX_CSR is found, the processor issues an error and aborts. If more than one AUX_CSR file is present, a warning is issued, but the processor proceeds with the latest one. Note that it does so on the basis of the date of the last modification of the file. The file name is stored in `csrFileName`.

The AUX_CSR file contains the latest characterization of spectral transmission of the two Fabry-Perot (see [AUX-CSR]) and of the Fizeau via the copy of the ISR wherefrom it is issued. The output for the Fabry-Pérot is composed of the 3 vectors fFP, TA and TB. They are of the same length and contain frequencies (in MHz) and transmission factors for the Fabry-Perot A and B. For the Fizeau, two vectors $fFiz$ and $TFiz$ are created and contain the frequencies (in MHz) and the transmission through the Fizeau.

Note that since fFP and $fFiz$ are in GHz in the AUX_CSR, a conversion to MHz is applied.

MATLAB Code

```
%--------------------------------------------------------------------------------
% Reads the AUX_CSR file currently valid in order to retrieve the transmission
% characteristics of the two FPs and the Fizeau.
files = dir(fullfile(auxdir,'*AUX_CSR_1B_*'));
if isempty(files),
                error('No AUX_CSR file found in the AUX directory.');
else
    nfiles = length(files);
    if nfiles > 1,
        warning('[GENERATEAUXCAL]: More than 1 AUX_CSR file in AUX directory.');
    end
    % Always take latest AUX_CSR file
```

```
    tfile = zeros(nfiles,1);
    for kfile = 1:nfiles,
        tfile(kfile) = datenum(files(kfile).date);
    end
    [xdum,kmax] = max(tfile);
    csrFileName = files(kmax).name;
    auxCSRData = readXml(fullfile(auxdir,csrFileName));
    [tstart,tstop] = UpdateValidityDates(csrFileName,0,0);

    results = auxCSRData.Earth_Explorer_File.Data_Block.Corrected_Spectral_Registration.
              List_of_Data_Set_Records.Data_Set_Record.
              List_of_CSR_Frequency_Steps.CSR_Frequency_Step;
    nfreq = length(results);

    fFP  = zeros(nfreq,1);
    TA   = zeros(nfreq,1);
    TB   = zeros(nfreq,1);

    for kfreq = 1:nfreq,
        fFP(kfreq) = 1000*str2num(results(kfreq).Laser_Freq_Offset{1});
        TA(kfreq)  = str2num(results(kfreq).Rayleigh_A_Response{1});
        TB(kfreq)  = str2num(results(kfreq).Rayleigh_B_Response{1});
    end

    results = auxCSRData.Earth_Explorer_File.Data_Block.Internal_Spectral_Registration….
              List_of_Data_Set_Records.Data_Set_Record.List_of_ISR_Results.ISR_Result;
    nfreq = length(results);
    fFiz  = zeros(nfreq,1);
    TFiz  = zeros(nfreq,1);
    for kfreq = 1:nfreq,
        fFiz(kfreq) = 1000*str2num(results(kfreq).Laser_Freq_Offset{1});
        TFiz(kfreq) = str2num(results(kfreq).Fizeau_Transmission{1});
    end
end
```

The transmission characteristics of the two FPs are re-sampled into vectors $fint$, $Taint$ and $TBint$. The purpose is to make sure the frequency resolution of the characterization is $param.df$ and the frequency excursion range is $[-param.FSRFP, +param.FSRFP]$. Note that the vector $fint$ must be centered about 0.

MATLAB Code

```
fint = param.Df :param.Df :1000*param.FSRFP ;
fint = [-fliplr(fint),0,fint];
nfint= length(fint);
Taint= interp1(fFP,TA,fint,'linear','extrap') ;
Tbint= interp1(fFP,TB,fint,'linear','extrap');
```

The Fizeau transmission vector $TFiz$ is also resampled to in a similar way into the $Tfizint$. For this it uses the Free Spectral Range of the Fizeau specified in the AUX_PAR_CL file.

MATLAB Code

```
%-------------------------------------------------------------------------------
% Resample Tfiz and extends the frequency range. This is done by using the known
% periodicity of the curve FRSFiz.
fwrap    = fint - 1000*round(0.001*fint / param.FSRFiz)*param.FSRFiz;
Tfizint  = interp1(fFiz,Tfiz,fwrap,'spline',NaN);
```

4.3 COMPUTATION OF FUNCTIONS C_1 TO C_4.

Functions C_1 to C_4 are computed by applying equation (5) and numerically integrating equation (3). The convolution products in (3) are approximated by a finite difference scheme similar to the one

developed for the computation of the AUX_RBC (see [AUX-RBC]). The normalizations introduced by equation (4) are applied afterwards.

The molecular spectrum can be either pure Rayleigh or Rayleigh-Brillouin. The model is selected via the RBC_Spec_Model setting in the AUX_PAR_CL file. For pure Rayleigh, the spectrum is computed by the $Rayleigh\_Spectrum$ function:

$$Rayleigh\_Spectrum(f, f_0, T, P) = \frac{1}{\sqrt{2\pi}\delta\nu(T)} \exp\left(-\frac{x^2}{2\delta\nu(T)^2}\right) \tag{11}$$

where

$$\delta\nu(T) = \frac{2}{\lambda}\sqrt{\frac{2k_bT}{M}} \tag{12}$$

with $k_B = 1.38\ 10^{-23} Js^{-1}$ the Boltzam's constant, and $M = 4.789\ 10^{-26} kg$ the mass of an air molecule

For Rayleigh-Brillouin, the function $Tenti\_Spectrum$ is used. It is based upon Witschas' model[1]:

$$\begin{aligned} &Tenti\_Spectrum(f, f_0, T, P) \\ &= \frac{A}{\sqrt{2\pi}\sigma_R}\exp\left(-\frac{x^2}{2\sigma_R^2}\right) + \frac{1-A}{2\sqrt{2\pi}\sigma_B}\exp\left(-\frac{(x-x_B)^2}{2\sigma_R^2}\right) \\ &+ \frac{1-A}{2\sqrt{2\pi}\sigma_B}\exp\left(-\frac{(x+x_B)^2}{2\sigma_R^2}\right) \end{aligned} \tag{13}$$

where

$$\begin{aligned} x &= \frac{f-f_0}{\delta\nu(T)} \\ x_B &= 0.80893 - 0.30208\times 0.10898^y \\ A &= 0.18526 * exp(-1.31255y) + 0.07103exp(-18.26117 * y) + 0.74421 \\ \sigma_R &= 0.70813 - 0.16366y^2 + 0.19132y^3 - 0.07217y^4 \\ \sigma_B &= 0.07845\ exp(-4.88663y) + 0.80400\ exp(-0.15003y) - 0.45142 \end{aligned} \tag{14a}$$

with

$$y = \frac{P}{2\pi\delta\nu(T)\eta} \tag{14b}$$

The parameter η in $Pa\ m^{-1}s^{-1}$ is the shear viscosity of air. It is temperature dependant

$$\eta = \eta_0 \sqrt{\frac{\left(\frac{T}{T_{ref}^B}\right)^3 \left(T_{ref}^B + T_{ref}^{shear}\right)}{T + T_{ref}^{shear}}}$$

with $\eta_0 = 1.846\ 10^{-5}\ Pa\ m^{-1}s^{-1}$, $T_{ref}^B = 300\ K$ and $T_{ref}^{shear} = 110.4\ K$

[1] B. Witschas, "Analytical model for Rayleigh–Brillouin line shapes in air," Appl. Opt. 50, 267–270 (2011).

MATLAB Code

```
%--------------------------------------------------------------------------------------------
% Frequencies for which the calibration functions must be computed
% The frequency step is df, the vector is centered about 0
fcal = param.df:param.df:500*param.USR;
fcal = [fliplr(fcal),0,-fcal];

%--------------------------------------------------------------------------------------------
% Sets the frequency vector fspec needed for the computation of the convolution products
% of Rayleigh spectra and transmission curves TA, TB and Tfiz.
fspec = param.df:param.df:1000*param.FSRFP+fcal(1));
fspec = [-fliplr(fspec),0,fspec];
nfspec = length(fspec);

% --------------------------------------------------------------------------------------------
% Computes calibration functions C2 = TA + TB and C3 = Tfiz
C2=interp1(fint,Taint+Tbint,fcal,'spline',NaN);
C3=interp1(fint,Tfizint,fcal,'spline',NaN);

%--------------------------------------------------------------------------------------------
% Computes C1 and C4 by convolving TA+TB and Tfiz with Rayleigh spectra
Spec = zeros(1,nfspec);
if strcmp(param.specMod,'GAUSS'),
   for kPcal=1:nPcal,
      for kTcal=1:nTcal,
         Spec = Rayleigh_Spectrum(0.001*fspec,0,Tcal(kTcal),Pcal(kPcal));
         for kfcal=1:nfcal,
            C1(kPcal,kTcal,kfcal) =
0.001*param.df*sum((TAint+TBint).*Spec(kfcal:kfcal+nfint-1));
            C4(kPcal,kTcal,kfcal) = 0.001*param.df*sum( TFizint
.*Spec(kfcal:kfcal+nfint-1));
         end
      end
   end
elseif strcmp(param.specMod,'TENTI'),
   for kPcal=1:nPcal,
      for kTcal=1:nTcal,
         Spec = Tenti_Spectrum(0.001*fspec,0,Tcal(kTcal),Pcal(kPcal));
         for kfcal=1:nfcal,
            C1(kPcal,kTcal,kfcal) =
0.001*param.df*sum((TAint+TBint).*Spec(kfcal:kfcal+nfint-1));
            C4(kPcal,kTcal,kfcal) = 0.001*param.df*sum( TFizint
.*Spec(kfcal:kfcal+nfint-1));
         end
      end
   end
else
   error('[GENERATECAL]: Spec model is either GAUSS or TENTI.');
end
%--------------------------------------------------------------------------------------------
% Normalisation of C1, C2, C3 and C4. C1 and C2 are divided by C1(1000hPa,300K,0MHz),
% while C3 and C4 are divided by C4(1000hPa,300K,0MHz)
if strcmp(param.specMod,'GAUSS'),
   Spec = Rayleigh_Spectrum(0.001*fspec,0,300,1000);
elseif strcmp(param.specMod,'TENTI'),
   Spec = Tenti_Spectrum(0.001*fspec,0,300,1000);
else
   error('[GENERATECAL]: Spec model is either GAUSS or TENTI.');
end
kfcal0 = 0.5*(nfcal+1);
C10 = 0.001*param.df*sum((Taint+Tbint).*Spec(kfcal0:kfcal0+nfint-1));
C40 = 0.001*param.df*sum(Tfizint      .*Spec(kfcal0:kfcal0+nfint-1));
C1=C1/C10;C2=C2/C10;C3=C3/C40;C4=C4/C40;
```

4.4 COMPUTATION OF K_{Ray} AND K_{Mie}.

The computation of K_{Ray} and K_{Mie} proceeds in five steps:

1. First, the AUX_MRC_1B file is read and the Mie frequency step geolocations (date, latitude, longitude, altitude, satellite range), normalized useful signals and scattering ratios are extracted. Note the validity of each frequency step is checked. The information is stored in the mrc structure. It contains the fields $freq$, $date$, lat, lon, alt, rng, $signal$, and $scatratio$.
2. Likewise, the AUX_RRC_1B file is read and Rayleigh frequency step geolocations (date, latitude, longitude, altitude, satellite range) and normalized useful signals are extracted. They are store in the rrc structure in fields $freq$, $date$, lat, lon, alt, rng, and $signal$. Note that frequencies and geolocations should be identical in the AUX_RRC_1B and AUX_MRC_1B files since both derive from the same AUX_IRC_1A product.
3. The AUX_MET_12 product(s) is (are) read and AUX_MET meteorological profiles and AUX_RRC_1B frequency steps are matched-up. If two AUX_MET_12 product are found, they are concatenated before match-up.
4. "Predicted" Mie and Rayleigh useful signals $\hat{S}_{Ray}(z)$ and $\hat{S}_{Mie}(z)$ are computed for all valid RRC/MRC frequency steps. A frequency step is valid when the corresponding information in the RRC and MRC files are declared valid and an AUX_MET_12 profile has been found by the match-up algorithm.
5. At last, the ratio of the average of the observed useful signals in the altitude range $[z_{min}, z_{max}]$ to the average of the "predicted" useful signals is formed.

4.4.1 Step 1: MRC information

MATLAB Code

```
%--------------------------------------------------------------------------------------------
% Read the MRC_1B file
files = dir(fullfile('./SCENARIOS',scenarioName,'*AUX_MRC_1B*.EEF'));
if isempty(files),
    error('[GENERATEAUXCAL]: No AUX_MRC_1B file found in the scenario directory.');
else
    if length(files) > 1,
        error('[GENERATEAUXCAL]:More than 1 MRC_1B file found in the scenario directory.');
    else
        mrcFileName = files(1).name;
        mrcData = readXml(fullfile('./SCENARIOS',scenarioName,mrcFileName));
        [tstart,tstop] = UpdateValidityDates(mrcFileName,tstart,tstop);
    end
end

% Data block
datanode = mrcData.Earth_Explorer_File.Data_Block.Auxiliary_Calibration_MRC.
           List_of_Data_Set_Records.Data_Set_Record;

% Frequency steps and validity
freqnode  = datanode.List_of_Frequency_Step_Results.Frequency_Step_Result;
mrc.nfreq = length(freqnode);
mrc.freq  = zeros(1,mrc.nfreq);
mrc.valid = zeros(2,mrc.nfreq);
for kfreq = 1:mrc.nfreq,
    mrc.freq(kfreq)  = str2double(freqnode(kfreq).Frequency_Offset{1});
    if strcmp(freqnode(kfreq).Frequency_Valid{1},'TRUE'),
        mrc.valid(1,kfreq) = 1;
    end
end
```

```
% Mie useful signal and scattering ratio
mrc.signal    = zeros(24,mrc.nfreq);
mrc.scatratio = zeros(24,mrc.nfreq);
for kfreq=1:mrc.nfreq,
    mrc.signal(:,kfreq)    = sscanf(freqnode(kfreq).Normalized_Useful_Signal{1},'%f');
    mrc.scatratio(:,kfreq) = sscanf(freqnode(kfreq).Mie_Scattering_Ratio{1},'%f');
end

% Geolocations
geonode = datanode.List_of_Frequency_Step_Geolocations.Frequency_Step_Geolocation;
mrc.date = zeros(1,mrc.nfreq);
mrc.lat  = zeros(1,mrc.nfreq);
mrc.lon  = zeros(1,mrc.nfreq);
mrc.alt  = zeros(25,mrc.nfreq);
mrc.rng  = zeros(25,mrc.nfreq);

basetime=datenum('01-Jan-2000 00:00:00');
for kfreq = 1:mrc.nfreq,
    mrc.date(1,kfreq) =
                86400*(datenum(sscanf(geonode(kfreq).Start_of_Observation_Time_Last_BRC{1},…
                                          'UTC=%4d-%2d-%2dT%2d:%2d:%2d',[1 6]))-basetime);
    mrc.lat(1,kfreq)   = 0.000001*str2double(geonode(kfreq).Latitude_of_DEM_Intersection{1});
    mrc.lon(1,kfreq)   = 0.000001*str2double(geonode(kfreq).Longitude_of_DEM_Intersection{1});
    mrc.alt(:,kfreq)   = sscanf(geonode(kfreq).Altitude{1},'%f');
    mrc.rng(:,kfreq)   = sscanf(geonode(kfreq).Satellite_Range{1},'%f');
end

% Check there are enough valid frequency steps in the MRC_1B file
numValid = sum(mrc.valid(1,:));
if numValid < param.minFreqStepsValid,
    error('[GENERATEAUXCAL] : Not enough valid responses in MRC file.');
end
```

4.4.2 Step 2: RRC information

MATLAB Code

```
%--------------------------------------------------------------------------------------------
% Read RRC 1B file
files = dir(fullfile('./SCENARIOS',scenarioName,'*AUX_RRC_1B*.EEF'));
if isempty(files),
    error('No AUX_RRC_1B file found in the scenario directory.');
else
    if length(files) > 1,
        error('More than 1 AUX_RRC_1B file found in the scenario directory.');
    else
        rrcFileName = files(1).name;
        rrcdata = readXml(fullfile('./SCENARIOS',scenarioName,rrcFileName));
        [tstart,tstop] = UpdateValidityDates(mrcFileName,tstart,tstop);
    end
end

datanode = rrcdata.Earth_Explorer_File.Data_Block.
           Auxiliary_Calibration_RRC.List_of_Data_Set_Records.Data_Set_Record;

% Frequency steps and validity
freqnode  = datanode.List_of_Frequency_Step_Results.Frequency_Step_Result;
rrc.nfreq = length(freqnode);
rrc.freq  = zeros(1,rrc.nfreq);
rrc.valid = zeros(1,rrc.nfreq);
for kfreq = 1:rrc.nfreq,
    rrc.freq(kfreq)   = str2double(freqnode(kfreq).Frequency_Offset{1});
    if strcmp(freqnode(kfreq).Frequency_Valid{1},'TRUE'),
        rrc.valid(1,kfreq) = 1;
    end
end

% Rayleigh useful signal and scattering ratio
rrc.signal    = zeros(24,rrc.nfreq);
for kfreq=1:rrc.nfreq,
    rrc.signal(:,kfreq) = sscanf(freqnode(kfreq).Normalized_Useful_Signal{1},'%f');
```

```
end

% Geolocations
geonode = datanode.List_of_Frequency_Step_Geolocations.Frequency_Step_Geolocation;
rrc.date = zeros(1,rrc.nfreq);
rrc.lat  = zeros(1,rrc.nfreq);
rrc.lon  = zeros(1,rrc.nfreq);
rrc.alt  = zeros(25,rrc.nfreq);
rrc.rng  = zeros(25,rrc.nfreq);
rrc.geoid = zeros(1,rrc.nfreq);

basetime=datenum('01-Jan-2000 00:00:00');
for kfreq = 1:rrc.nfreq,
    rrc.date(1,kfreq) =
                86400*(datenum(sscanf(geonode(kfreq).Start_of_Observation_Time_Last_BRC{1},…
                                                'UTC=%4d-%2d-%2dT%2d:%2d:%2d',[1 6]))-basetime);
    rrc.lat(1,kfreq)   = 0.000001*str2double(geonode(kfreq).Latitude_of_DEM_Intersection{1});
    rrc.lon(1,kfreq)   = 0.000001*str2double(geonode(kfreq).Longitude_of_DEM_Intersection{1});
    rrc.alt(:,kfreq)   = sscanf(geonode(kfreq).Altitude{1},'%f');
    rrc.rng(:,kfreq)   = sscanf(geonode(kfreq).Satellite_Range{1},'%f');

    ngdsep = length(geonode(kfreq).List_of_Geoid_Separations.Geoid_Separation);
    temp = geonode(kfreq).List_of_Geoid_Separations.Geoid_Separation(ceil(ngdsep/2));
    rrc.geoid(kfreq)   = sscanf(temp{1},'%f');
end

% Check there are enough valid frequency steps in the MRC_1B file
numValid = sum(rrc.valid(1,:));
if numValid < param.minFreqStepsValid,
    error('[GENERATEAUXCAL] : Not enough valid responses in RRC file.');
end

%-------------------------------------------------------------------------------------------
% Check RRC and MRC have same date, lat and lon
if any(abs(mrc.date-rrc.date)>0),
    error('[UPDATECSRFROMRRC]: MRC and RRC dates are not identical');
end

if any(abs(mrc.lat-rrc.lat)>0),
    error('[UPDATECSRFROMRRC]: MRC and RRC latitudes are not identical');
end

if any(abs(mrc.lon-rrc.lon)>0),
    error('[UPDATECSRFROMRRC]: MRC and RRC longitudes are not identical');
end
```

4.4.3 Step 3: AUX_MET data

The AUX_MET file corresponding to the IRC operation is read. The information is stored in the structure $auxMet$. The structure contains a one-dimensional array of substructures Obs. There are as many elements $Obs(kobs)$ as there are profiles in the AUX_MET file. Each element is composed of the 5 fields named $date$, lat, lon, alt, $pressure$ and $temperature$. The fields alt, $pressure$ and $temperature$ are vectors of equal size, they define the vertical profiles of temperature and pressure provided by the numerical weather forecast system. The fields $date$, lat, and lon are scalars; they are the date, latitude and longitude of the profile.

If two AUX_MET products are available for the IRC, they are concatenated in the $auxMet$ structure.

The second operation consists in matching-up AUX_MET profiles and AUX_RRC_1B frequency steps. The procedure is copied from the L2B processor and the AUX_CSR updater, it is fully described in §8.3.1 of [AUX-CSR]. It is tuned by the parameter settings $time\_max$ and $range\_max$ (see Table 1).

The result of the match-up operation is the structure array $metData$. Its length is equal to the number of valid observations in the AUX_RRC_1B file. The element $metData(kobs)$ is empty if there is no meteorological profile matching-up the kobs-th RRC_1B frequency step, otherwise it contains the arrays $altitude$, $temperature$ and $pressure$ defining the corresponding meteorological profile.

The number of valid RRC/MRC frequency steps are determined. If one of them is less than $minFreqStepValid$, the processor stops and reports an error.

MATLAB Code

```
%-------------------------------------------------------------------------------------------
% Read AUX_MET data files for the given scenario.
Files=dir(fullfile('./SCENARIOS',scenarioName,'AE_TEST_AUX_MET_12*.DBL'));
if isempty(files),
    error('No AUX_MET data for specified scenario.');
else
    nfiles = length(files);
    if nfiles>2,
        error('[UPDATECSRFROMRRC]: More than two AUX_MET files for the scenario.');
    else
        if nfiles==1;
            metFileName{1} = files.name;
            metFileName{2} = 'unused';

        else
            metFileName{1} = files(1).name;
            metFileName{2} = files(2).name;
        end
    end
end

kprof = 0;
for kfile = 1:nfiles,
       auxMetData = ReadBinaryDataCAL(…
                           fullfile(pwd,'SCENARIOS',scenarioName,metFileName{kfile}),…
                                                              'Geo_ADS2','Met_MDS2');

    % Make sure the DS2 is not empty.
    If auxMetData.Num_Records_in_DS2 < 1,
        error(fprintf('[UPDATECSRFROMRRC]: No data in DS2 of AUX_MET file %d.',kfile));
    end

    % Read aux met information in DS2
    for krec = 1:auxMetData.Num_Records_in_DS2,

        % Increment profile index
        kprof = kprof+1;

        % Read geolocation data for the current profile. The date is equal to
        % the number of seconds elapsed since 01 Januray 2000 00:00 and is
        % truncated to the an integer.
        geonode = auxMetData.Geolocation_ADS2.DSR(krec);
        auxMet.Obs(kprof).date  = 86400*double(geonode.AMD_DateTime(1))+…
                                  double(geonode.AMD_DateTime(2));
        auxMet.Obs(kprof).lat   = 1e-6*double(geonode.AMD_Latitude);
        auxMet.Obs(kprof).lon   = 1e-6*double(geonode.AMD_Longitude);
        % Read altitude shift between Geoide and WGS84 models
        auxMet.Obs(kprof).zg    = double(geonode.AMD_zg);

        % Read the Met data profile, that is pressure and temperature versus altitude.
        metnode = auxMetData.Met_MDS2.DSR(krec).List_of_Profile_Data;
        auxMet.Obs(kprof).altitude    = 0.01*double([metnode.AMD_znom]);
        auxMet.Obs(kprof).pressure    = 0.01*double([metnode.AMD_pnom]);
        auxMet.Obs(kprof).temperature = 0.01*double([metnode.AMD_T]);
    end
end

%-------------------------------------------------------------------------------------------
```

```
% Matches up RRC Observation and Met profiles. Uses the algorithm 17reviously developed for
the
% AUX_CSR updater
metData=matchAMD4RRC(rrc,auxMet,param);

% Fill line 2 of validity matrix.
For kfreq = 1:mrc.nfreq,
    if isempty(metData(kfreq).altitude),
        mrc.valid(2,kfreq) = 0;
    else
        mrc.valid(2,kfreq) = 1;
    end
end

% Check there are enough MRC frequency steps with valid aux met profile
numValid = sum(mrc.valid(2,:));
if numValid < param.minFreqStepsValid,
    error('[GENERATEAUXCAL]: Not enough RRC obs with valid atm profile.');
end

% Check there are enough MRC and RRC frequency steps with all information needed to proceed
numValid = sum(prod(mrc.valid,1).*rrc.valid(1,:));
if numValid < param.minFreqStepsValid,
    error('Not enough IRC obs with all information needed.');
end
```

4.4.4 Step 3: Prediction of the Mie and Rayleigh useful signals

The prediction of the Mie and Rayleigh useful signals $\hat{S}_{Mie}$ and $\hat{S}_{Ray}$ is carried out in a way similar to what is done by the AUX_CSR updater (see §8.3.2 in [AUX-CSR]).

The initial step is the definition of a fine-resolution, vertical grid:

$$zref = Zref\_Max : -Dz\_Ref : Zref\_Min \tag{15}$$

The vertical resolution Dz_Ref is a tunable parameter. It is currently set to $Dz\_Ref = 25$ (in meters). The vector $zref$ must start with the highest altitude. The parameters $Zref\_Max$ and $Zref\_Min$ (both in meters) are tunable. The interval $[Zref\_Min, Zref\_Max]$ must encompass the altitude range over which the IRC is performed.

Then, for each valid IRC frequency step (index $iobs = 1, \dots, nobs$), the vertical pressure and temperature AUX_MET profiles are interpolated to the altitudes $zref$:

$$Tref(:, iobs) = interp(metData.altitude(iobs) + rrc.geoid(iob), metData.temperature(iobs), zref) \tag{16a}$$

$$Pref(:, iobs) = interp(metData.altitude(iobs) + rrc.geoid(iobs), metData.pressure(iobs), zref) \tag{16b}$$

The arrays $Tref$ and $Pref$ are used in order to predict the molecular backscatter $bmol$ and extinction $amol$ coefficients along the lidar line-of-sight.

$$bmol = 2.351319\ 10^{-6} \frac{Pref}{Tref}$$
$$amol = \frac{8\pi}{3} * bmol \tag{17}$$

This prediction is is done with the $molOptProp$ function. The processor then computes the expected strength of the Rayleigh return according to the standard lidar equation:

$$molStrength(izref, iobs) = \frac{bmol(izref, iobs)}{range(izref)^2} \exp\left(-2 \sum_{k=2}^{izref} amol(k, iobs) dRange(k)\right) \quad (18)$$

Here, $izref$ is the index of the current altitude, and $range(izref)$ is the distance along the line-of-sight between the lidar and the altitude $zref(izref)$, and $dRange(k) = range(k) - range(k-1)$. It is obtained by interpolating the ranges of MRC height bins (reported in the Geolocation_ADS of the MRC_1B file) to the altitudes $zref$:

$$range = inter(mrc.alt(:, iobs), mrc.range(:, iobs), zref) \quad (19)$$

The calibration functions C_1 and C_4 are then interpolated to the MRC/RRC frequency steps and the meteorological conditions:

$$C_1^{ref} = interp3(Tcal, Pcal, fcal, C_1, Tref, Pref, fref)$$
$$C_4^{ref} = interp3(Tcal, Pcal, fcal, C_4, Tref, Pref, fref) \quad (20)$$

At last, equations (6) are computed with the interpolated values of C_1 and $C_4$and $\hat{S}_{Ray}(z)$ and $\hat{S}_{Mie}(z)$ are integrated over the height-bins of each observation.

MATLAB Code

```
% Reference altitudes for computing molecular return strength. The vertical resolution is set
% by dzref. Beware, the vector ZREF must contain altitudes in descending order.
Zref=param.zrefmax :-param.dzref :param.zrefmin ;
nzref=length(zref);

% Select IRC observations with all necessary information
jfreq = find(prod(mrc.valid,1).*rrc.valid(1,:));

% Check again the minimum number is reached, but this should have already been done reviously.
If isempty(jfreq),
    error('No valid IRC information');
else
    numFreqValid = length(jfreq);
    if numFreqValid < param.minFreqStepsValid,
        error('[GENERATEAUXCAL]: Not enough IRC obs with required information.');
    end
end

% Interpolation of auxMet data to reference altitudes and computation of molecular
% backscatter and extinction + strength of Rayleigh return.
Pref        = zeros(nzref,numFreqValid);
Tref        = zeros(nzref,numFreqValid);
Range       = zeros(nzref,numFreqValid);
molStrength = zeros(nzref,numFreqValid);

for kfreq = 1:numFreqValid,
    Pref(:,kfreq) = interp1(metData(jfreq(kfreq)).altitude+rrc.geoid(jfreq(kfreq)),…
                                    metData(jfreq(kfreq)).pressure,zref,'spline','extrap');
    Tref(:,kfreq)  = interp1(metData(jfreq(kfreq).altitude+rrc.geoid(jfreq(kfreq)),…
                                  metData(jfreq(kfreq)).temperature,zref,'spline','extrap');
    Range(:,kfreq) = interp1(mrc.alt(:,jfreq(kfreq),1),…
                                     mrc.rng(:,jfreq(kfreq),1),zref','spline','extrap');
end

[amol,bmol] = molOptProp(Pref,Tref);
molStrength = bmol.*exp(-2*cumsum(amol.*[zeros(1,numFreqValid);diff(Range)])./(Range.*Range);
```

```
% Interpolation of C1 and C4 to Pref and Tref.
fref  = repmat(mrc.freq(jfreq)',nzref,1);
C1ref = interp3(Tcal,Pcal,fcal,C1,Tref,Pref,fref,'linear',NaN);
C4ref = interp3(Tcal,Pcal,fcal,C4,Tref,Pref,fref,'linear',NaN);

% Integration of molecular strength return in Rayleigh and Mie height bins
PredRaySignal = zeros(24,numFreqValid);
PredMieSignal = zeros(24,numFreqValid);

for kfreq=1:numFreqValid,
    PredRaySignal(:,jfreq(kfreq)) = param.dzref*…
                              diff(interp1(zref,cumsum(C1ref(:,kfreq).*molStrength(:,kfreq)),…
                                                    rrc.alt(:,jfreq(kfreq),1),'spline',NaN));
    PredMieSignal(:,jfreq(kfreq)) = param.dzref*…
                              diff(interp1(zref,cumsum(C4ref(:,kfreq).*molStrength(:,kfreq)),…
                                                    mrc.alt(:,jfreq(kfreq),2),'spline',NaN));
end
```

4.4.5 Step 4: Computation of K_{ray} and K_{mie}

The processor selects all the Mie and Rayleigh heigh-bins

1. Tagged valid,
2. with a top altitude less than Z_Max and a bottom altitude more than Z_Min,
3. with a scattering ratio less than the threshold Max_Mie_Scat defined in AUX_PAR_CL (see Table 1).

For Rayleigh height-bins, the third condition above must be met for all the intersecting Mie height-bins.

The total number of photons in selected Mie and Rayleigh height-bins is controlled. If it is more than a predefined threshold ($param.minSignalLevel$), then the ratio of the number of photocounts in Mie and Rayleigh channel by the predicted signal levels (equation 9) is calculated. If the threshold is not met, the processor returns an invalid value (-999.999).

MATLAB Code

```
% Computation of Kray and Kmie
% First, the scattering ratios in Mie height-bins are used to determine whether there is
% a significant amount of Mie scattering in Mie and Rayleigh height-bins. For Mie, this
% amounts to detecting the scattering ratios above a predefined threshold. For Rayleigh
% bins, we must look for the Mie bins that contain the Rayleigh bins and see if one %
contains a significant amount of Mie scattering.
MieScatInMieFlag = zeros(24,irc.nFreq);
MieScatInRayFlag = zeros(24,irc.nFreq);
for kfreq = 1:irc.nFreq,

    for kbin = 1:24,

        % Mie bins. Test the scatratio in the bins.
        If scatratio(kbin,kfreq) < 0,
            MieScatInMieFlag(kbin,kfreq) = -1;
        elseif scatratio(kbin,kfreq) > param.MieScatThresh,
            MieScatInMieFlag(kbin,kfreq) = 0;
        else
            MieScatInMieFlag(kbin,kfreq) = 1;
        end
```

```
        % Rayleigh bins. More complicated. For each Rayleigh bin, find the corresponding
        % Mie bins, and then discard the Rayleigh bin is any of the Mie bin inside is
        % invalid or above the threshold.
        ibinTop = find(irc.alt(:,kfreq,2) >= irc.alt(kbin  ,kfreq,1));
        ibinBot = find(irc.alt(:,kfreq,2) <= irc.alt(kbin+1,kfreq,1));
        if isempty(ibinTop) | isempty(ibinBot),
            MieScatInRayFlag(kbin,kfreq) = -1;
        else
            BinSet = max(ibinTop):(min(ibinBot)-1);
            if any(scatratio(BinSet,kfreq) > param.MieScatThresh),
                MieScatInRayFlag(kbin,kfreq) = 0;
            elseif any(scatratio(BinSet,kfreq) < 0),
                MieScatInRayFlag(kbin,kfreq) = -1;
            else
                MieScatInRayFlag(kbin,kfreq) = 1;
            end
        end
    end % Loop kbin
end % Loop kfreq


iAltRay = find((rrc.alt(1:24,jfreq)<param.zmax)&(rrc.alt(2:25,jfreq)>param.zmin)&…
               (PredRaySignal>0)&(MieScatInRayFlag(:,jfreq)==1));
iAltMie = find((mrc.alt(1:24,jfreq)<param.zmax)&(mrc.alt(2:25,jfreq)>param.zmin)&…
               (PredMieSignal>0)&(MieScatInMieFlag(:,jfreq)==1));

signal = rrc.signal(:,jfreq);
if sum(ObsRaySignal(iAltRay)) > param.minSignalLevel,
    Kray = sum(signal(iAltRay))/sum(PredRaySignal(iAltRay));
else
    Kray = -999.999;
    warning('[GENERATEAUXCAL]: Not enough signal for estimation of Kray.');
end

signal = mrc.signal(:,jfreq);
if sum(signal(iAltMie)) > param.minSignalLevel,
    Kmie = sum(signal(iAltMie))/sum(PredMieSignal(iAltMie));
else
    Kmie = -999.999;
    warning('[GENERATEAUXCAL]: Not enough signal for estimation of Kmie.');
end
```

4.5 VALIDITY PERIOD

The validity period of the output AUC_CAL product are determined from the validity dates of the input products AUX_CSR_1B, AUX_ISR_1B, AUX_MRC_1B and AUX_RRC_1B. The determination is done by the function $UpdateValidityDates$ called when these input products are read.

5. Input and output files

5.1 GENERAL PRODUCT STRUCTURE

The general structure of the files below complies with the general product structure presented in section 3 of [L1B-IODD]. Since the fixed headers of all AEOLUS files are identical as far as their size and structure are concerned, we omit their description in the following sections. The detail of the file structure thus starts with the Specific Product Header before proceeding to the Data Block.

5.2 INPUT FILES

5.2.1 AUX_PAR_CL

The AUX_PAR_CL file contains the parameter settings of the AUX_CAL generator. It is an XML file named

`AE_CCCC_AUX_PAR_CL_yyyymmddThhmmss_yyyymmddThhmmss_vvvv.EEF`

where `CCCC` is either `TEST` or `OPER`, and the groups `yyyymmddThhmmss` define the start and end of sensing time, and `vvvv` is the version number.

The overall size of the file is 4Ko.

Name	Description / Comment	Unit	Type	Size (XML)
Specific_Product_Header	Root tag.		Structure	26 0 27
Sph_Descriptor	Specific Product Header descriptor: ASCII string describing the product		String	16 27 18
Spare_1			Spare	11 0 0
List_of_Dsds	See Table 3 for list of Data Set Descriptors.		Structure	308
Total size in bytes				**433**

Table 2: Specific Product Header of the AUX_PAR_CL file

Name	Description / Comment	Unit	Type	Size (XML)
CAL_PAR_GADS	DSD for the result of the AUX_CSR		G	267
Total size in bytes				**267**

Table 3: Data Set Descriptors contained in the List_of_Dsds in the SPH.

Name	Description / Comment	Unit	Type	Size (XML)
CAL_Parameters	Data Set containing the calibration parameters. See Table 5.		Structure	17 911 18
Total size in bytes				**946**

Table 4: Content of the Data Block of the AUX_PAR_CL file

Name	Description / Comment	Count	Type	Size (XML)
List_of_Data_Set_Records	Data Set containing the calibration parameters. See Table 6.	1	Structure	37 846 28
Total size in bytes				**911**

Table 5: Content of the CAL_Paramaters GADS.

Name	Description / Comment	Unit	Type	Size (XML)
Data_Set_Record	Structure of the unique Data Set Record in CAL_Parameters Data Set. See Table 7 for a description.		Structure	18 848 19
Total size in bytes				**846**

Table 6: Content of the List_of_Data_Set_Records of the CAL_Parameters GADS.

Name	Description / Comment	Unit	Type	Size (XML)
Instrument	Structure containing several instrumental parameters that characterize the instrument. See Table 8.		Structure	13 118 14
Ref_Grid	Structure containing the parameters that define the vertical grid used for computing AUX_CAL constants and functions. See Table 9.		Structure	11 101 12
Atm_Grid	Structure that defines the pressure and temperature values for which calibration constants are computed. See Table 10.		Structure	11 207 12
Matchup	Structure containing the L1B versus AUX_MET matchup thresholds. See Table 11.		Structure	10 71 11
RBC_Spec_Model	Switch selecting the molecular spectral model. It can be either 'GAUSS' or 'TENTI'.		String	16 5 18
Thresholds	Structure containing several thresholds used by the AUX_CAL generator. See Table 12.		Structure	13 191 14
Total size in bytes				**848**

Table 7: Content of a Data_Set_record in the CAL_Parameters GADS

Name	Description / Comment	Unit	Type	Size (XML)
USR	Useful Spectral Range	GHz	Fado1.3	16 5 7
FSRFP	Free Spectral Range of the Fabry-Perot	GHz	Fado2.3	18 6 9
FSRFiz	Free Spectral Range of the Fizeau	GHz	Fado1.3	19 5 10
Df	Frequency resolution	MHz	intAus	15 2 6
Total size in bytes				**118**

Table 8: Content of the Instrument structure of a CAL_Parameters Data Set Record.

Name	Description / Comment	Unit	Type	Size (XML)
Dz_ref	Vertical resolution of the grid	m	IntAus	17 2 10
Zref_Min	Bottom altitude of the grid	m	IntAul	19 5 12
Zref_max	Top altitude of the grid	m	IntAul	19 5 12
Total size in bytes				**101**

Table 9: Content of the Ref_Grid structure of a CAL_Parameters Data Set Record.

Name	Description / Comment	Unit	Type	Size (XML)
Tcal_Min	Minimun, maximum temperatures and temperature step defining the vector of temperatures for which calibration constants and function are to be computed.	K	IntAus	19 3 12
Tcal_Max		K	IntAul	19 3 12
Tcal_Stp		K	IntAul	19 1 12
Pcal_Min	Same as above for the pressure	hPa	IntAus	21 2 12
Pcal_Max		hPa	IntAul	21 4 12
Pcal_Stp		hPa	IntAul	21 2 12
Total size in bytes				**207**

Table 10: Content of the Atm_Grid structure of a CAL_Parameters Data Set Record.

Name	Description / Comment	Unit	Type	Size (XML)
Range_Max	Maximum distance tolerated between a L1B and an AUX_MET profile.	km	IntAus	21 3 13
Time_Max	Maximum time difference tolerated between a L1B and an AUX_MET profile.	s	IntAul	19 3 12
Total size in bytes				**71**

Table 11: Content of the Matchup structure of a CAL_Parameters Data Set Record.

Name	Description / Comment	Unit	Type	Size (XML)
Min_Freq_Steps_Valid	Minimum number of valid frequency steps required for the completion of the AUX_CAL generation		IntAus	22 2 24
Z_min	Bottom altitude of the range considered for the estimation of K_{ray} and K_{mie}.	m	IntAul	16 5 9
Z_max	Top altitude of the range considered for the estimation of K_{ray} and K_{mie}.	m	IntAul	16 5 9
Min_Signal_Level	Minimum value of the normalized useful signal accumulated in selected height-bins for Kray and Kmie estimates		IntAul	18 10 20
Max_Mie_Scat	Maximum value of the scattering ratio allowed for a height bin to be included in the computation of K_{Ray} and K_{Mie}.		Fado1.3	14 5 16
Total size in bytes				**191**

Table 12: Content of the Thresholds structure of a CAL_Parameters Data Set Record.

5.2.2 AUX_CSR_1B

The content of the file is defined in [AUX-CSR].

5.2.3 AUX_ISR_1B

The content of the file is defined in [L1B-IODD].

5.2.4 AUX_MRC_1B

The content of the file is defined in [L1B-IODD].

5.2.5 AUX_RRC_1B

The content of the file is defined in [L1B-IODD].

5.2.6 AUX_MET

The content of the file is defined in [L2BC-IODD].

5.3 OUTPUT FILE

The AUX_CAL generator outputs a single product that contains all the calibration constants and functions computed by the processor. Due to the large number of data to be stored, the product is in a binary format. According to AEOLUS rules, it is thus made of two separate files: a header file in XML (terminated by the .HDR extension), and a binary data block file (terminated by a .DBL extension). Both share the same name:

`AE_CCCC_AUX_CAL_L2_yyyymmddThhmmss_yyyymmddThhmmss_vvvv`

where `CCCC` is either `TEST` or `OPER`, and the groups `yyyymmddThhmmss` define the start and end of sensing time, and `vvvv` is the version number.

The structure of the product is detailed below. Sizes are given for a number of pressure levels $Num\_P = 105$, a number of temperature levels $Num\_T = 201$, a number of frequency steps $Num\_fd = 61$, and the transmission functions T_A, T_B and T_{Fiz} characterized for $Num\_FP = 877$ frequency steps.

Tag Name	Content Description	Unit	Type	Size (KVT)	Size (XML)
Specific_Product_Header	Root tag for XML format only.		Structure	N/A	53
Sph_Descriptor	"AEOLUS_AUX_CAL_L2A_SPH" (SPH descriptor ASCII string describing the product.)		String	46	56
Spare_1			String	41	11
Ref_CAL_Suite	String indicating whether the AUX_CAL file was generated operationally or as part of tests		String	37	36
Num_P	Number of pressure levels (4 digits)		IntAus	13	20
Num_T	Number of temperature levels		IntAus	13	20
Num_Fd	Number of frequency steps for the characterization of the calibration constants and functions		IntAus	14	22
Num_FP	Number of frequency steps for the characterization of FP and Fizeau transmissions		IntAus	14	22
Spare_2			String	41	11
P_Min	Minimum pressure considered for the characterization of calibration functions (6 digits)	Pa	IntAus	17	32
P_Max	Maximum pressure considered for the characterization of calibration functions (6 digits)	Pa	IntAul	17	32
T_Min	Minimum temperature considered for the characterization of calibration functions (5 digits)	10-2K	IntAus	20	35
T_Max	Maximum temperature considered for the characterization of calibration functions (5 digits)	10-2K	IntAus	20	35
Fd_Min	Minimum frequency considered for the characterization of calibration functions (10 digits + sign)	MHz	IntAl	24	40
Fd_Max	Maximum frequency considered for the characterization of calibration functions (10 digits + sign)	MHz	IntAl	24	40
Spare_3				41	11
List_of_DSDs	List of the 7 DSDs reported in the file. Six of them are references to products input to the AUX_CAL generator. See Table 14.		Structure	2016	2222
Total size for KVT and XML SPH in bytes:				**2398**	**2698**

Table 13: Specific product header of the AUX_CAL_L2 product.

Num.	Data Set Descriptor Name	Content Description	Data Set Type	Update Frequency
1	CAL_ADS	Data Set containing the calibration constants and functions	A	Every time a CSR or an IRC is performed
2	PAR_ADS	Parameter settings.	R	N/A
3	CSR_ADS	Reference to the AUX_CSR used for the generation of the AUX_CAL product	R	Every time an ISR or an IRC is performed
4	MRC_ADS	Reference to the MRC_1B file	R	Every time an IRC is performed. The RRC and PRC products must be derived from the same IRC
5	RRC_ADS	Reference to the RRC_1B file	R	
6	MT1_ADS	Reference to AUX_MET product 1	R	Variable
7	MT2_ADS	Reference to AUX_MET product 2. If this second AUX_MET file is missing, this field is set to 'unused'.	R	Variable

Table 14: List of Data Set Descriptors in the AUX_CAL_L2 product SPH.

Tag Name	Content Description	Unit	Type	Size
P_Grid	Vector of size Num_P of the pressure levels for which calibration functions C_1 and C_4 are given.	Pa	IntAul	420
T_Grid	Vector of size Num_T of the temperature levels for which calibration functions C_1 and C_4 are given. Size given for a vector of size 201.	10-2K	IntAus	402
Fd_Grid	Vector of size Num_Fd of the frequency steps for which calibration functions C_1 to C_4 are given. Size given for a vector of size 61.	MHz	IntAd	488
F_FP	Vector of size Num_FP of the frequency steps for which the transmission curves of the FPs A and B and the Fizeau are determined.	MHz	IntAd	7016
Ta_FP	Transmission curve of Fabry Perot A. Size given for a vector F_FP of 877 elements.		FAdoxy	7016
Tb_FP	Transmission curve of Fabry Perot B. Size given for a vector F_FP of 877 elements.		FAdoxy	7016
Tmie_FP	Transmission curve of the Fizeau. Size given for a vector F_FP of 877 elements.		FAdoxy	7016
Cal_Coeff_Ray	Structure containing C_1 and C_4 as a function of pressures P_Grid, temperatures T_Grid and frequencies fd_Grid. See Table 16.		Structure	20598488
Cal_Coeff_Mie	Structure containing C_2 and C_3 as a function of the frequencies fd_Grid. See Table 20.		Structure	984

Tag Name	Content Description	Unit	Type	Size
Total size for binary MDSR in bytes:				**20628846**

Table 15: Content of CAL_ADS.

Tag Name	Content Description	Unit	Type	Size
Kray	Kray coefficient		Fadoxy	8
List_of_Cal_Coeff_Ray_P	List of Rayleigh coefficients C_1 and C_4 for a given pressure level.			20598480
Total size for binary MDSR in bytes:				**20598488**

Table 16: Content of Cal_Coeff_Ray.

Tag Name	Content Description	Unit	Type	Size
List_of_Cal_Coeff_Ray_PT	List of Rayleigh coefficients C_1 and C_4 for a given pressure and temperature level.			196176
Total size for binary MDSR in bytes:				**196176**

Table 17: Content of List_of_Cal_Coeff_Ray_P.

Tag Name	Content Description	Unit	Type	Size
List_of_Cal_Coeff_Ray_PTFd	List of Rayleigh calibration coefficients for a given set of pressure and temperature.			976
Total size for binary MDSR in bytes:				**976**

Table 18: Content of List_of_Cal_Coeff_Ray_PT.

Tag Name	Content Description	Unit	Type	Size
C1	Calibration coefficient C_1.		FAdoxy	8
C4	Calibration coefficient C_4.		FAdoxy	8
Total size for binary MDSR in bytes:				**16**

Table 19: Content of List_of_Cal_Coeff_Ray_PTfd.

Tag Name	Content Description	Unit	Type	Size
Kmie	Kmie coefficient		Fadoxy	8
List_of_Cal_Coeff_Mie_Fd	List of Mie coefficients C_2 and C_3 as a function of the frequency Fd			976
Total size for binary MDSR in bytes:				**984**

Table 20: Content of List_of_Cal_Coeff_Mie.

Tag Name	Content Description	Unit	Type	Size
C2	Calibration coefficient C_2			8
C3	Calibration coefficient C_3			8
Total size for binary MDSR in bytes:				**16**

Table 21: Content of List_of_Cal_Coeff_Mie_fd.