

Changes to **SME** structures for version 5

Jeff A. Valenti¹, Nikolai E. Piskunov²

February 23, 2014

¹valenti@stsci.edu

²nikolai.piskunov@physics.uu.se

Document History

Issue	Revision	Date	Author	Comment
D	1	2014-02-13	JV	Created initial version

Contents

1	Introduction	3
1.1	Document conventions	3
1.2	SME version numbers	3
1.3	Document context	4
2	New software utilities	5
2.1	Update an SME structures	5
2.2	Convert an output structure to an input structure	5
2.3	Check an atmosphere structure	6
3	Atmosphere fields in SME structures	7
3.1	Atmosphere fields in version 3.1 SME structures	7
3.2	Atmosphere fields in version 5 SME structures	8
3.3	Relationship between version 3.1 and version 5 fields	10
3.4	Presence of fields in input and output structures	13
4	Atmosphere grids	15
4.1	Grid sizes and extents	15
4.2	Grid fields	17
5	Notes for SME software developers	19
5.1	SME procedures affected by structure updates	19
5.2	Atmosphere handling in the GUI	20

Chapter 1

Introduction

1.1 Document conventions

SME in the normal font refers to the Spectroscopy Made Easy software package, which includes the GUI and IDL procedure interfaces.

SME in the typewriter font refers to an IDL structure variable named **SME**, which contains an SME calculation request (“input structure”) and may also contain the calculation results (“output structure”).

1.2 SME version numbers

Table 1.1 describes the types of version numbers used by SME.

Table 1.1: Three types of version numbers in SME

Version type	Description
SME Library	Version number of the compiled external library that handles equation of state and radiative transfer calculations. Library files are named <code>sme_synth*.so</code> . <i>Example:</i> 3.93, June 2013
SME Package	Version number of the IDL software that implements the GUI, the solver, and the interface with the external library. The package version number is the revision number of the code in an SVN repository. <i>Example:</i> 351

Table 1.1 – *Continued*

Version type	Description
SME Structure	Version number of the specification for SME data structures. We increase the structure version number when old structures must be updated to be compatible with the current code. This document describes such a transition from version 3.1 to version 5. <i>Example: 3.1</i>

All three version numbers are printed when the procedure `sme_main` executes. For example:

```
IDL> restore, 'example.inp'
IDL> sme_main, sme
SME Library version:  3.93, June 2013
SME Package version:  351
SME Structure version: 5.0
```

1.3 Document context

This document describes differences between SME structure version 3.1 and SME structure version 5. Structure version 3.1 was the last official version prior to structure version 5. The version number of an SME structure is stored in `SME.VERSION` and is distinct from the SME package number.

SME package version 407 was the first to create and handle version 5 structures. We aspire to make each new release of the SME software backwards compatible with existing **SME** structures. Please contact us if you encounter an old structure that the current software is unable to process. On the other hand, SME package version 406 and earlier cannot process version 5 structures. In this situation, you should update your SME software.

We increment the SME structure version when an **SME** structure generated by older SME software must be updated before it can be processed properly by the latest SME software. When the SME software encounters an old **SME** structure version, the code immediately updates the **SME** structure to the current structure version. Thereafter we assume that the **SME** structure obeys the current specification. This makes the code cleaner and easier to read. When the SME software writes an **SME** structure, the code always writes an SME structure with the current structure version.

Chapter 2

New software utilities

2.1 Update an SME structures

Starting with package version 407, SME includes a procedure `sme_struct_update` that updates an existing SME structure to the current specification. SME calls `sme_struct_update` immediately after reading an SME structure from disk, so users generally do not have to call the routine directly. However, users can call the routine directly if desired, as in this example:

```
restore, 'example_v31.inp'
sme_struct_update, sme, errstr
if errstr ne '' then message, errstr
save, sme, file='example_v5.inp'
```

2.2 Convert an output structure to an input structure

The procedure `sme_struct_update` can also be used to strip all output fields from an SME structure and from the `SME.ATMO` substructure, leaving only input fields. This is useful for converting the results from a previous calculation into a new SME request. If necessary, the structure version is updated to the current specification before stripping output fields.

As discussed later in [3.4](#), the value of `sme.method` affects whether certain `sme.atmo` fields are allowed in an input `sme` structure. Users must set the desired value of `sme.method` before calling `sme_struct_update` with the `/input` flag. Here is an example:

```
restore, 'example_v31.out'
sme.method = 'embedded'
sme_struct_update, sme, errstr, /input
if errstr ne '' then message, errstr
save, sme, file='example_v5.inp'
sme_main, sme
save, sme, file='example_v5.out'
```

2.3 Check an atmosphere structure

The procedure `sme_check_atmo_struct` tests whether `SME.ATMO` or a standalone `ATMO` structure is consistent with the current atmosphere structure specification. `SME` calls this procedure after reading an `SME` structure from disk and updating it to the current specification (see section ??). `SME` also calls this procedure after creating a new `ATMO` structure.

Some users develop their own software to create `SME` input structures, rather than using the `SME` GUI. Users should call `sme_check_atmo_struct` to validate that `SME.ATMO` obeys the current atmosphere structure specification. Here is an example.

```
atmo = {method:'grid', source:'krz2.sav', depth:'RHOX'}  
errstr = sme_check_atmo_struct(atmo=atmo)  
if errstr ne '' then message, errstr
```

Chapter 3

Atmosphere fields in SME structures

3.1 Atmosphere fields in version 3.1 SME structures

Prior to structure version 5, `SME.ATMO` was a `[5,NDEP]` or `[6,NDEP]` array, depending on whether the atmosphere was plane parallel or spherical. Additional atmosphere information was stored in separate SME structure fields, as described in Table 3.1.

Table 3.1: Description of atmosphere fields in version 3.1 SME structures

Field	Description
<code>SME.ATMO_PRO</code>	Name of IDL procedure that generates an atmosphere, usually by interpolating in a grid. <i>Example:</i> 'interpkrz2'
<code>SME.ATMOGRID_FILE</code>	Name of IDL save file containing an atmosphere grid in the form expected by <code>interp_atmo_grid.pro</code> . <i>Example:</i> 'marcs2010t02p.sav'
<code>SME.ATM_WLSTD</code>	Wavelength for continuum optical depth scale. <i>Units:</i> Å, <i>Default value:</i> 5000 Å
<code>SME.ATM_TYPE</code>	Indicates whether depth scale in <code>SME.ATMO[0,*]</code> is continuum optical depth or mass column. A value of 'SPH' implies a spherical model with a mass column depth scale. <i>Allowed values:</i> 'TAU', 'RHOX', or 'SPH'.
<code>SME.N_ATM</code>	Set to 1 when atmosphere is provided in SME input structure. No grid interpolation in this case.

Table 3.1 – *Continued*

Field	Description
SME.ATMO	Array containing atmosphere properties at each depth. ATMO[0,*] Depth scale (<code>tau</code> or <code>rhox</code>) ATMO[1,*] Temperature [K] ATMO[2,*] Electron number density [cm^{-3}] ATMO[3,*] Atomic number density [cm^{-3}] ATMO[4,*] Mass density [g cm^{-3}] ATMO[5,*] Height (spherical atmospheres only) [cm]
SME.ATM_TEFF	Effective temperature of the input atmosphere, when SME.N_ATM=1. No effect on output. <i>Units:</i> K
SME.ATM_GRAV	Surface gravity of an input atmosphere, when SME.N_ATM=1. No effect on output. <i>Units:</i> $\log(\text{cm s}^2)$
SME.ATM_RADIUS	Stellar radius where height is defined to be zero in a spherical atmosphere. <i>Units:</i> cm
SME.ATM_FILE	Name of file that contained input atmosphere. No effect on output.
SME.FINAL_ATMO	Analog of SME.ATMO when atmosphere is an output, rather than an input.

3.2 Atmosphere fields in version 5 SME structures

The most significant change in version 5 structures is the consolidation of separate atmosphere fields into a single SME.ATMO field with subfields described in Table 3.2.

Table 3.2: Description of atmosphere fields in version 5.0 SME structures

Field	Description
SME.ATMO	Atmosphere structure with fields as described below.
SME.ATMO.METHOD	The method SME will use to obtain the atmosphere. <i>Allowed values:</i> 'grid', 'embedded', or 'routine'

Table 3.2 – Continued

Field	Description
SME.ATMO.SOURCE	Source of atmosphere data. When METHOD is 'grid' this field contains the name (but not the path) of the IDL save file containing the atmosphere grid. <i>Example: 'marcs2010t02p.sav'</i>
SME.ATMO.DEPTH	Type of depth scale to use for radiative transfer. <i>Allowed values: 'TAU' or 'RHOX'</i>
SME.ATMO.INTERP	Type of depth scale to use when interpolating a grid of atmospheres with <code>interp_atmo_grid</code> . <i>Allowed values: 'TAU' or 'RHOX'</i>
SME.ATMO.GEOM	Geometry of radiative transfer, which can be plane parallel or spherical. When METHOD is 'grid', this field may be a null string in an input structure. In this case SME will select geometry during grid interpolation. <i>Allowed values: 'PP', 'SPH', or ''</i>
SME.ATMO.WLSTD	Wavelength for continuum optical depth scale. <i>Units: Å, Default value: 5000 Å</i>
SME.ATMO.TAU	Continuum optical depth at each tabulated depth in the atmosphere.
SME.ATMO.RHOX	Mass column at each tabulated depth in the atmosphere. <i>Units: g cm⁻²</i>
SME.ATMO.TEMP	Temperature at each tabulated depth in the atmosphere. <i>Units: K</i>
SME.ATMO.XNE	Electron number density at each tabulated depth in the atmosphere. <i>Units: cm⁻³</i>
SME.ATMO.XNA	Atomic number density (including atomic components of molecules) at each tabulated depth in the atmosphere. <i>Units: cm⁻³</i>
SME.ATMO.RHO	Mass density at each tabulated depth in the atmosphere. <i>Units: g cm⁻³</i>
SME.ATMO.HEIGHT	Height above or below SME.ATMO.RADIUS at each tabulated depth in a spherical atmosphere. <i>Units: cm</i>

Table 3.2 – *Continued*

Field	Description
SME.ATMO.RADIUS	Stellar radius corresponding to HEIGHT of zero in a spherical atmosphere. <i>Units:</i> cm
SME.ATMO.TEFF	Effective temperature used to generate the atmosphere. Use SME.TEFF to specify effective temperature when interpolating in a grid. <i>Units:</i> K
SME.ATMO.LOGG	Surface gravity used to generate the atmosphere. Use SME.GRAV to specify surface gravity when interpolating in a grid. <i>Units:</i> $\log(\text{cm s}^{-2})$
SME.ATMO.MONH	[M/H] metallicity used to generate the atmosphere. Use SME.FEH to specify metallicity when interpolating in a grid.
SME.ATMO.ABUND	Elemental abundances (relative to the total number of atoms) used to generate the atmosphere. Use SME.ABUND to specify abundances when interpolating in a grid.
SME.ATMO.VTURB	Turbulent velocity used to generate the atmosphere. <i>Units:</i> km s^{-1}
SME.ATMO.LONH	Ratio of mixing length to pressure scale height (ℓ/H) used to generate the atmosphere.
SME.ATMO.OPFLAG	Flags that indicate whether to enable various opacity packages during the radiative transfer calculation.

3.3 Relationship between version 3.1 and version 5 fields

Table 3.3 gives the mapping between fields in a version 5 SME.ATMO structure and fields in an version 3.1 SME structure. For the atmosphere grid interpolation case, several fields (e.g., RADIUS) have no analog in the SME structure, but can exist in atmosphere grid structures. Table 3.3 uses ATMO_GRID as a generic name for atmosphere grid structures, but some grids have different names (e.g., KRZ2).

Table 3.3: Mapping between fields in version 3.1 and version 5

Version 5	Version 3.1	
	Grid case	Embedded case
SME.ATMO.METHOD	'grid'	'embedded'
SME.ATMO.SOURCE	(see table 3.4)	SME.ATM_FILE

Table 3.3 – *Continued*

Version 5	Version 3.1	
	Grid case	Embedded case
SME.ATMO.DEPTH	(see table 3.4)	SME.ATM_TYPE
SME.ATMO.INTERP	(see table 3.4)	SME.ATM_TYPE
SME.ATMO.GEOM	(see table 3.4)	SME.ATM_TYPE
SME.ATMO.WLSTD	ATMO_GRID.WLSTD	SME.ATM_WLSTD
SME.ATMO.TAU	SME.ATMO[0,*]	SME.ATMO[0,*]
SME.ATMO.RHOX	SME.ATMO[0,*]	SME.ATMO[0,*]
SME.ATMO.TEMP	SME.ATMO[1,*]	SME.ATMO[1,*]
SME.ATMO.XNE	SME.ATMO[2,*]	SME.ATMO[2,*]
SME.ATMO.XNA	SME.ATMO[3,*]	SME.ATMO[3,*]
SME.ATMO.RHO	SME.ATMO[4,*]	SME.ATMO[4,*]
SME.ATMO.HEIGHT	SME.ATMO[5,*]	SME.ATMO[5,*]
SME.ATMO.RADIUS	ATMO_GRID.RADIUS	SME.ATM_RADIUS
SME.ATMO.TEFF	ATMO_GRID.TEFF	SME.ATM_TEFF
SME.ATMO.LOGG	ATMO_GRID.LOGG	SME.ATM_GRAV
SME.ATMO.MONH	ATMO_GRID.MONH	(no analog)
SME.ATMO.ABUND	ATMO_GRID.ABUND	(no analog)
SME.ATMO.VTURB	ATMO_GRID.VTURB	(no analog)
SME.ATMO.LONH	ATMO_GRID.LONH	(no analog)
SME.ATMO.OPFLAG	ATMO_GRID.OPFLAG	(no analog)

Table 3.4: Mapping of `SME.ATMO_PRO` in version 3.1 to `SME.ATMO` fields in version 5

Version 3.1	Version 5			
<code>SME.ATMO_PRO</code>	<code>SME.ATMO.SOURCE</code> ¹	<code>SME.ATMO.DEPTH</code> ²	<code>SME.ATMO.INTERP</code> ²	<code>SME.ATMO.GEOM</code>
'interp_atmo_grid'	'atlas12.sav'	<code>ATMO_GRID.MODTYP</code>	<code>ATMO_GRID.MODTYP</code>	<code>ATMO_GRID.SPHERE</code>
'interpatlas12'	'atlas12.sav'	'TAU'	'TAU'	'PP'
'interpatlas12_rhox'	'atlas12.sav'	'RHOX'	'RHOX'	'PP'
'interpkrz2'	'atlas12.sav'	'RHOX'	'RHOX'	'PP'
'interpkrz3'	'krz3.sav'	'TAU'	'TAU'	'PP'
'interpmarcs2008'	'marcs2008t2.sav'	'RHOX'	'RHOX'	'PP'
'interpmarcs2010p'	'marcs2010t02p.sav'	'RHOX'	'RHOX'	'PP'
'interpmarcs2010s'	'marcs2010t02s.sav'	'RHOX'	'RHOX'	'SPH'
'interpmarcs2012'	'marcs2012.sav'	'TAU'	'TAU'	'PP'
'interpmarcs2012_rhox'	'marcs2012.sav'	'RHOX'	'RHOX'	'PP'

¹ If set, `SME.ATMOGRID_FILE` overrides the value of `SME.ATMO.SOURCE` given in the table above.

² If set, `SME.ATM_TYPE` overrides the value of `SME.ATMO.DEPTH` and `SME.ATMO.INTERP` given in the table above.

3.4 Presence of fields in input and output structures

Table 3.5 indicates whether structure fields are always, sometimes, or never present in input and in output SME.ATMO structures. Table 3.5 distinguishes between two cases, depending on whether SME.ATMO.METHOD is 'grid' or 'embedded'. In the embedded case, fields in output structure are simply a copy of the same field in the input structure. Table 3.6 explains the meanings of all relevant combinations of “Always”, “Maybe”, “Never”, and “Input”.

Table 3.5: Presence of fields in input and output structures

Field	Grid Case		Embedded Case	
	Input	Output	Input	Output
SME.ATMO.SOURCE	Always	Input	Maybe	Input
SME.ATMO.DEPTH	Maybe	Always	Always	Input
SME.ATMO.INTERP	Maybe	Always	Maybe ¹	Input
SME.ATMO.GEOM	Maybe	Always	Always	Input
SME.ATMO.WLSTD	Never	Maybe ²	Maybe ²	Input
SME.ATMO.TAU	Never	Maybe ³	Maybe ³	Input
SME.ATMO.RHOX	Never	Maybe ³	Maybe ³	Input
SME.ATMO.TEMP	Never	Always	Always	Input
SME.ATMO.XNE	Never	Always	Always	Input
SME.ATMO.XNA	Never	Always	Always	Input
SME.ATMO.RHO	Never	Always	Always	Input
SME.ATMO.HEIGHT	Never	Maybe ⁴	Maybe ⁴	Input
SME.ATMO.RADIUS	Never	Maybe ⁴	Maybe ⁴	Input
SME.ATMO.TEFF	Never	Maybe	Maybe	Input
SME.ATMO.LOGG	Never	Maybe	Maybe	Input
SME.ATMO.MONH	Never	Maybe	Maybe	Input
SME.ATMO.ABUND	Never	Maybe	Maybe	Input
SME.ATMO.VTURB	Never	Maybe	Maybe	Input
SME.ATMO.LONH	Never	Maybe	Maybe	Input
SME.ATMO.OPFLAG	Never	Maybe	Maybe	Input

¹ The only allowed value is a null string.

² If TAU is present, then WLSTD must be present.

³ At least one of TAU or RHOX must be present. Both may be present.

⁴ If HEIGHT is present, then RADIUS must be present.

Table 3.6: Explanation of “Always”, “Maybe”, and “Never” flags

Input	Output	Explanation
Always	Input	Field is <i>always</i> present in input and output SME structures. SME <i>always</i> copies the input value to the output structure.
Maybe	Input	Field <i>may</i> be present in input SME structures. If present, SME copies the input value to the output structure. Otherwise, the field will <i>not</i> be in the output structure.
Maybe	Always	Field <i>may</i> be present in input SME structures. If present, SME copies the input value to the output structure. Otherwise, SME creates and populates the field in the output structure.
Maybe	Maybe	Field may be present in input SME structures. If present, SME copies the input value to the output structure. Otherwise, SME <i>may</i> create and populate the field in the output structure.
Never	Always	Field is <i>never</i> present in input SME structures. SME <i>always</i> creates and populates the field in the output structure.
Never	Maybe	Field is <i>never</i> present in input SME structures. SME <i>may</i> create and populate the field in the output structure.

For atmosphere grid interpolation, the input `SME.ATMO` structure must specify `METHOD='grid'` and `SOURCE`. The output `SME.ATMO` structure will contain `METHOD`, `SOURCE`, `DEPTH`, `INTERP`, `GEOM`, (`TAU` or `RHOX`), `TEMP`, `XNE`, `XNA`, and `RHO`. If `TAU` is present, then `WLSTD` will be present. If `GEOM='SPH'`, then `HEIGHT` and `RADIUS` will be present. The remaining parameters will be present, if they appear in the atmosphere interpolation grid.

For an embedded atmosphere, the input `SME.ATMO` structure must specify `METHOD='embedded'`, `DEPTH`, `GEOM`, (`TAU` or `RHOX`), `TEMP`, `XNE`, `XNA`, and `RHO`. If `TAU` is present, then `WLSTD` must be present. If `GEOM='SPH'`, then `HEIGHT` and `RADIUS` must be present. The `OPFLAG` field may be present and will affect the result. The remaining input fields are metadata that do not affect the result. The `INTERP` field is allowed, but if present the value must be a null string. This makes it easy to use an output `SME.ATMO` from a grid interpolation case as input for an embedded atmosphere case. The output `SME.ATMO` structure is identical to the input structure. No fields are added or removed. All input values are preserved without modification.

Chapter 4

Atmosphere grids

4.1 Grid sizes and extents

Table 4.1 lists all the atmosphere grids that have appeared in an SME release.

Column 1 contains a concise grid code (e.g., G01), used to cross-reference data in subsequent tables in this chapter. These grid codes have no meaning and should never be referenced outside this document. Instead, users should reference grids by the name of the IDL save file, which is given in column 2 of the table and in the `SME.SOURCE` field.

Column 3 gives the name of the IDL structure variable in the IDL save file. Many previous versions of SME stored the grid in an IDL structure named `KRZ2`. New atmosphere grids should store the grid in an IDL structure named `ATMO_GRID`.

Column 6 indicates whether the IDL save file for each grid also contains an introductory description of the grid in a variable named `ATMO_GRID_INTRO`. If yes, then table 4.2 gives the text of the grid description.

Table 4.1: Sizes and extents of atmosphere interpolation grids

ID	Grid File	Grid Var	N_{atm}	N_{dep}	Ver	Intro	T_{eff}		$\log g$		$[M/H]$	
							Min	Max	Min	Max	Min	Max
G01	atlas12.sav	ATMO_GRID	7611	72	4.0	no	3500	50000	0.0	5.0	-5.0	1.0
G02	atlas9_vmic0.0.sav	ATMO_GRID	5456	288	4.0	yes	4000	10000	2.0	5.0	-1.0	1.0
G03	atlas9_vmic2.0.sav	ATMO_GRID	7752	288	4.0	yes	4000	20000	2.0	5.0	-1.0	1.0
G04	krz2.sav	KRZ2	7611	72	2.1	no	3500	50000	0.0	5.0	-5.0	1.0
G05	krz3.sav	KRZ3	4289	72	3.0	no	3500	50000	0.0	5.0	-4.0	-0.1
G06	ll_vmic2.0.sav	ATMO_GRID	42266	72	4.0	yes	4500	22000	2.5	5.0	-0.8	0.8
G07	marcs2.sav	KRZ2	920	72	2.0	no	3000	7000	3.0	5.0	-1.5	0.8
G08	marcs2008t2.sav	KRZ2	2290	72	2.0	no	2500	8000	3.0	5.5	-5.0	1.0
G09	marcs2010t02p.sav	KRZ2	1862	72	3.0	no	2500	8000	3.0	5.5	-2.0	1.0
G10	marcs2010t02s.sav	KRZ2	2313	72	3.0	no	2500	8000	-0.5	3.5	-2.0	1.0
G11	marcs2012.sav	ATMO_GRID	4289	72	4.0	no	2500	8000	-0.5	5.0	-5.0	1.0
G12	marcs2012t02.sav	KRZ2	4306	72	3.0	no	2500	8000	-0.5	5.0	-5.0	1.0
G13	marcs2012t02cooldwarfs.sav	KRZ2	1267	72	3.0	no	2500	4000	3.0	5.5	-5.0	1.0

Table 4.2: Explanatory text for some atmosphere interpolation grids

Grid	Description
G02	This grid of models was produced in Vienna using ATLAS9 code of R. Kurucz and published in Heiter et al. 2002, A&A 392, 619. Some of the models were later recomputed. The grid contains 5456 models and covers the range of T_{eff} from 4000 K to 10000 K with a step of 200 K. The range of $\log g$ is 2.0 to 5.0 with a step of 0.2, and $[\text{m}/\text{H}]$ of ± 1.0 , ± 0.5 , ± 0.3 , ± 0.2 , ± 0.1 and 0. Microturbulence is 0 km s^{-1} and the mixing length parameter is 0.9. One model ($T_{\text{eff}}=4000 \text{ K}$, $\log g=5$, $[\text{m}/\text{H}]=-0.5$) strange problem was found with electron number densities this were replaced by calculations using SME EOS. τ_{5000} was computed using SME EOS and continuous opacity package.
G03	(Same as grid G02)
G06	This grid of models was produced Denis Shulyak using his line-by-line code derived from ATLAS9. The grid was published in Shulyak et al. 2004, A&A 428, 993. The grid contains 42266 models and covers the range of T_{eff} from 4500 K to 10000 K with a step of 100 K and from 10000 K to 22000 K with a step of 250 K. The range of $\log g$ is 2.5 to 5.0 with a step of 0.1, and $[\text{m}/\text{H}]$ from -0.8 to $+0.8$ with a step of 0.1. Microturbulence is 2.0 km s^{-1} . Convection is included for models up to 9000 K and it follows the Canuto-Mazzitelli prescription.

4.2 Grid fields

Table 4.3 indicates with an ‘×’ symbol which fields exist in each atmosphere grid structure. Table 4.1 maps grid identifiers (e.g. G01) to IDL save files (e.g. flatlas12.savfl) and other grid properties.

All grids have a mass column (RHOX) depth scale. Some grids also have a TAU depth scale. Grids with RADIUS, HEIGHT, and SPHERE include spherical atmospheres.

Table 4.3: Structure fields present in atmosphere interpolation grids

[illegible]

Table 4.3 – *Continued*

Field	G01	G02	G03	G04	G05	G06	G07	G08	G09	G10	G11	G12	G13
XNA	×	×	×	×	×	×	×	×	×	×	×	×	×
RHO	×	×	×	×	×	×	×	×	×	×	×	×	×
NDEP	×	×	×	×	×	×	×	×	×	×	×	×	×
TAU	×	×	×		×	×					×		
RADIUS									×	×	×	×	×
HEIGHT									×	×		×	×
SPHERE									×	×		×	×

Grid G11 (`marcs2012.sav`) contains only plane parallel atmospheres, despite the existence of the RADIUS field.

Chapter 5

Notes for SME software developers

5.1 SME procedures affected by structure updates

Table 5.1 lists the SME procedures that should call `sme_struct_update` because they operate on SME structures provided by users, either on the command line or from a disk file.

Table 5.1: SME procedures that should call `sme_struct_update`

Procedure	Use cases
GUI/sme_rdlin.pro	User selects “Line Data” → “OK” in the GUI. User wants to reuse line data from an input or output structure on disk.
GUI/sme_observations.pro	User selects “Observations” → “Read existing SME structure file” in the GUI. User wants to reuse a wavelength scale, observed spectrum, and associated uncertainties from an input or output structure on disk.
GUI/sme_useout.pro	User selects “Examine” → “Read SME Structure from Disk” in the GUI. User may want to examine calculation results in an output structure on disk. User may want to create a new input structure based on parameters in an input or output structure on disk.
sme_main.pro	User selects “Jobs” → “Submit SME Batch Job” in the GUI, which invokes <code>sme_main</code> from <code>sme_sme.pro</code> . User executes <code>run_next.inp</code> from the IDL prompt. User calls <code>sme_main</code> from the IDL prompt or from user-written code. In all cases, user wants to perform the SME calculation specified in an input structure on disk.
tools/export_model.pro	User calls <code>export_model</code> from the IDL prompt or from user-written code. User wants to create a .krz disk file from an output structure on disk.

Table 5.1 – *Continued*

Procedure	Use cases
<code>tools/cont_corr.pro</code>	User calls <code>cont_corr</code> from the IDL prompt or from user-written code. User wants to use the Wallace et al. (2011) solar atlas to obtain a continuum correction for the observed spectrum in an input or output structure on disk.
<code>tools/port_mask.pro</code>	User calls <code>port_mask</code> from the IDL prompt or from user-written code, passing filenames rather than structure variables. User wants to create update the observation mask in one input or output structure, based on the mask in another structure.

In principle, all of these procedures should call `sme_struct_update`, but only the procedures in **magenta** are affected by the transition to structure version 5. The `cont_corr` tool is affected because `SME.RESOL` can now be a vector. The remaining procedures in **magenta** are affected by the changes in atmosphere fields.

Users can pass an **SME** structure directly to `sme_main.pro`, so this routine must call `sme_struct_update`. Given this decision, there is no need for `sme_sme.pro` and `run_next_inp.pro` to call `sme_struct_update` because these two routines simply pass a restored **SME** structure to `sme_main`.

5.2 Atmosphere handling in the GUI

The GUI should handle the following atmosphere specification use cases.

- Manually create a new atmosphere specification
 1. Start new IDL session. Invoke `sme` procedure to start GUI.
 2. Select “Controls” → “Model Atmosphere”.
 3. Choose one of the following options. Test each option separately.
 - (a) Select “Grid Name”. Select specific grid from corresponding pulldown menu.
 - (b) Select “Single File”. Specify `.krz` or `.out` file in corresponding text box.
 - (c) Select “Routine Name”. Specify routine name in corresponding text box.
 4. Specify other options, as needed.
 5. Specify “Jobs” → “Review Current SME Request” to review atmosphere specification.
 6. Select “Jobs” → “Save Request as SME Input File” to write **SME** input structure to disk.
- Adopt an atmosphere specification from existing **SME** structure
 1. Start new IDL session. Invoke `sme` procedure to start GUI.

2. Select “Examine” → “Read SME structure from Disk”.
 3. Specify other options, as needed.
 4. Specify “Jobs” → “Review Current SME Request” to review atmosphere specification.
 5. Select “Jobs” → “Save Request as SME Input File” to write SME input structure to disk.
- Adopt parameters from existing SME structure, then update atmosphere specification
 1. Start new IDL session. Invoke `sme` procedure to start GUI.
 2. Select “Examine” → “Read SME structure from Disk”.
 3. Select “Controls” → “Model Atmosphere”.
 4. Choose one of the following options. Test each option separately.
 - (a) Select “Grid Name”. Select specific grid from corresponding pulldown menu.
 - (b) Select “Single File”. Specify `.krz` or `.out` file in corresponding text box.
 - (c) Select “Routine Name”. Specify routine name in corresponding text box.
 5. Specify other options, as needed.
 6. Specify “Jobs” → “Review Current SME Request” to review atmosphere specification.
 7. Select “Jobs” → “Save Request as SME Input File” to write SME input structure to disk.
 - Switch between different atmosphere specifications without restarting IDL
 1. Start new IDL session. Invoke `sme` procedure to start GUI.
 2. Select “Controls” → “Model Atmosphere”.
 3. Choose one of the following options.
 - (a) Select “Grid Name”. Select specific grid from corresponding pulldown menu.
 - (b) Select “Single File”. Specify `.krz` or `.out` file in corresponding text box.
 - (c) Select “Routine Name”. Specify routine name in corresponding text box.
 4. Specify “Jobs” → “Review Current SME Request” to review atmosphere specification.
 5. Repeat previous two steps for different methods and sources, checking result each time.

Note: In the “Controls” → “Model Atmosphere” widget, the Routine Name method should not offer deprecated `interp.pro` routines as options. For grid interpolation, users should select the Grid Name method instead. The Routine Name method is a hook for non-interpolation routines, e.g. a hypothetical routine that calls Atlas9 with the current abundance distribution.*