

LA-UR-

11-00075

Approved for public release;
distribution is unlimited.

Title: EOSPAC USER'S MANUAL: VERSION 6.2 (U)

Author(s): David A. Pimentel

Intended for: EOSPAC Users



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

January 5, 2011

EOSPAC User's Manual: Version 6.2

EOSPAC USER'S MANUAL: VERSION 6.2

LA-UR-

Artifact ID: EOSPAC6-02-01



David A. Pimentel (HPC-1)
January 5, 2011

This program was prepared by Los Alamos National Security, LLC at Los Alamos National Laboratory (LANL) under contract No. DE-AC52-06NA25396 with the U.S. Department of Energy (DOE). All rights in the program are reserved by the DOE and Los Alamos National Security, LLC. Permission is granted to the public to copy and use this software without charge, provided that this Notice and any statement of authorship are reproduced on all copies. Neither the U.S. Government nor LANS makes any warranty, express or implied, or assumes any liability or responsibility for the use of this software.

CONTENTS

1	INTRODUCTION	1
2	BASIC THEORY AND MODELS	1
2.1	NOMENCLATURE	1
2.2	ENTROPY	3
2.3	ION EOS MODELS	3
2.3.1	<i>Ideal Gas Model</i>	4
2.3.2	<i>Cowan Model</i>	5
2.3.3	<i>Number Proportional Model</i>	8
2.4	ADDITIONAL THERMODYNAMIC QUANTITIES	9
2.4.1	<i>Identities</i>	10
2.4.2	<i>Sound speed</i>	10
2.4.3	<i>Isentropic Compressibility</i>	11
2.4.4	<i>Isothermal Compressibility</i>	11
2.4.5	<i>Gruneisen Coefficient</i>	11
2.4.6	<i>Specific heats</i>	11
3	GENERAL INTERFACE DESCRIPTION	14
3.1	USAGE EXAMPLES	15
3.1.1	<i>Serial Case</i>	15
3.1.2	<i>Parallel Case</i>	16
4	CONVENTIONS	17
4.1	DATA ORGANIZATION	17
4.2	ROUTINE NAMES	18
4.3	CONSTANT IDENTIFIER NAMES	19
4.4	DATA TYPES	19
5	SETUP MATERIAL DATA	19
5.1	DATA LOCATIONS	20
5.1.1	<i>Environment-variable-defined and default search paths</i>	20
5.1.2	<i>Ordered File Names List Creation</i>	21
5.1.2.1	<i>Index file</i>	22
5.1.2.2	<i>Default file name list</i>	24
5.1.3	<i>Ordered File Names List Example</i>	24
5.2	DATA ORGANIZATION	26
5.3	ROUTINES AND PARAMETERS	27
5.3.1	<i>eos_CreateTables</i>	27
5.3.2	<i>eos_DestroyAll</i>	28
5.3.3	<i>eos_DestroyTables</i>	28
5.3.4	<i>eos_GetMaxDataFileNameLength</i>	29
5.3.5	<i>eos_GetPackedTables</i>	30
5.3.6	<i>eos_GetPackedTablesSize</i>	30
5.3.7	<i>eos_LoadTables</i>	31
5.3.8	<i>eos_SetDataFileName</i>	32
5.3.9	<i>eos_SetPackedTables</i>	33
5.4	C/C++ LANGUAGE BINDINGS	34
5.5	FORTRAN LANGUAGE BINDINGS	35

6	INTERPOLATE MATERIAL DATA	37
6.1	DATA ORGANIZATION	37
6.2	ROUTINES AND PARAMETERS	37
6.2.1	<i>eos_CheckExtrap</i>	37
6.2.2	<i>eos_Interpolate</i>	39
6.2.3	<i>eos_Mix</i>	40
6.3	C/C++ LANGUAGE BINDINGS	42
6.4	FORTRAN LANGUAGE BINDINGS	43
7	MISCELLANEOUS INFORMATION ROUTINES.....	44
7.1	ROUTINES AND PARAMETERS	44
7.1.1	<i>eos_GetErrorCode</i>	44
7.1.2	<i>eos_GetErrorMessage</i>	44
7.1.3	<i>eos_GetTableCmnts</i>	45
7.1.4	<i>eos_GetTableInfo</i>	45
7.1.5	<i>eos_GetVersion</i>	46
7.1.6	<i>eos_GetVersionLength</i>	46
7.1.7	<i>eos_ResetOption</i>	47
7.1.8	<i>eos_SetOption</i>	47
7.2	C/C++ LANGUAGE BINDINGS	48
7.3	FORTRAN LANGUAGE BINDINGS	49
8	SELECTED NUMERIC DETAILS.....	50
8.1	CUSTOM SMOOTHING AND INTERPOLATION	50
8.2	FORCED DATA MONOTONICITY	51
8.3	EXTENDED PRECISION IS DISABLED	52
8.4	MASS FRACTION DATA INTERPOLATION.....	53
8.5	NUMERICAL INTEGRATION	54
9	USAGE EXAMPLES	54
9.1	C HOST CODE EXAMPLE	54
9.2	C++ HOST CODE EXAMPLE	59
9.3	FORTRAN 77 HOST CODE EXAMPLE	64
9.4	FORTRAN 90 HOST CODE EXAMPLE	69
10	TECHNICAL SUPPORT INFORMATION.....	74
11	ACKNOWLEDGEMENTS	74
12	REFERENCES	75
APPENDIX A.	TABLE TYPE IDENTIFIER NAME MNEMONIC CONVENTIONS	
APPENDIX B.	TABLE TYPE DEFINITIONS GROUPED BY CATEGORY AND SORTED BY IDENTIFIER NAME	
APPENDIX C.	TABLE TYPE DEFINITIONS CROSS REFERENCED TO THOSE OF EOSPAC VERSION 5	

- APPENDIX D. SETUP PHASE OPTION FLAG DEFINITIONS**
- APPENDIX E. DATA INFORMATION PARAMETERS**
- APPENDIX F. INTERPOLATION PHASE OPTION FLAG DEFINITIONS**
- APPENDIX G. ERROR CODE DEFINITIONS**

1 INTRODUCTION

The EOSPAC utility package is a collection of interface routines, which can be used to access the SESAME data library and perform various data adjustments and interpolations on the SESAME data. The SESAME data library contains both thermodynamic (e.g., equation of state) and transport coefficients (e.g., opacity and conductivity), and it is described in reference 1. Note, for simplicity, the term EOS (equation of state) used herein includes both thermodynamic variables and transport coefficients. The EOSPAC utility package is designed to be used by physics codes (henceforth "host codes") written in multiple languages and on multiple platforms.

The remainder of this manual is organized into several sections. Section 2 provides a general overview of basic theory and models implemented within EOSPAC. Section 3 provides a general overview of how to use the EOSPAC interface library. Section 4 discusses conventions such as data organization and routine names. Sections 5 through 7 describe the public interfaces of EOSPAC in detail. Section 8 provides details related to some selected numerical features of EOSPAC. Section 9 gives examples for using the interface routines described in sections 5 through 7. Section 10 provides technical support contact information. Section 11 contains a brief set of acknowledgments. Finally, section 12 contains a list of referenced documents. Appendices list the Table Type Definitions, the Option Flag Definitions, the Information Flag Definitions, and the Error Code Definitions.

2 BASIC THEORY AND MODELS

SESAME typically contains EOS and Vaporization data, Melt & Shear Modulus data, Opacity data and Conductivity data [1]. Where EOS data is missing from SESAME, EOSPAC will often attempt to calculate it. In some cases, the host code can determine the models used to calculate EOS data.

2.1 Nomenclature

The variable nomenclature below is used throughout this section:

α Intrinsic Helmholtz free energy

C_{eVK} Electron-volt to Kelvin conversion factor (11604.85 K/eV)

c_p	Constant-pressure specific heat
c_v	Constant-volume specific heat
η	Electron degeneracy parameter
F	Fermi integral
Γ	Gruneisen coefficient
h	Intrinsic enthalpy
\hbar	Reduced Planck constant
K_s	Isentropic compressibility
K_T	Isothermal compressibility
k	Boltzman constant
κ	Ratio of specific heats
M	Average atomic mass
m	Mass
p	Total pressure
p_i	Ion pressure
p_c	Cold curve pressure (at $T = 0$)
N_i	Ion number density
ρ	Density
s	Intrinsic Entropy
R	Universal Gas Constant (8.314472e-03 kJ/K/mol)
T	Temperature
T_i	Ion temperature (eV)
T_D	Debye temperature (eV)
T_M	Lindemann melting temperature (eV)
u	Intrinsic Total internal energy

- u_i Intrinsic Ion internal energy
- v Intrinsic volume ($v = \frac{1}{\rho}$)
- Z Free electrons per ion

It is important to note that the intrinsic variables used in this section are lowercase, but are typically uppercase throughout the remainder of this document (specifically in the appendices). The upper case variants are a nomenclature artifact used to improve the readability of the mnemonics in which they are used. If questions arise regarding the units of a given quantity, then one should assume they are consistent with the documented SESAME data units [1].

2.2 Entropy

Entropy is an example of data, not stored within SESAME, which is simple to calculate using equation (2.1) if both the internal energy and Helmholtz free energy data are available.

$$a = u - Ts \tag{2.1}$$

If only the internal energy data is available, as is the case with older EOS data, then equations (2.2) and (2.3) are used to calculate entropy and equation (2.1) is subsequently used to calculate the Helmholtz free energy data.

$$s = \int_0^T \frac{1}{T} \frac{du}{dT} dT = \frac{u}{T} + \int_0^T \frac{u}{T^2} dT \tag{2.2}$$

$$s|_{T=0} = u|_{T=0} = \frac{u}{T}|_{T=0} = 0 \tag{2.3}$$

This integral form avoids the numerical sensitivities of other differential forms, which are discussed further in section 8.5.

2.3 Ion EOS Models

Other models are available to calculate EOS data corresponding to SESAME subtables [1]. These analytical models include the Ideal Gas Model, the Cowan Model and the Number Proportional Model. These models are used to create two-

temperature¹ EOS data by subtracting the analytically-calculated data from SESAME's tabulated total EOS data. Due to cautionary guidance [2], experimentation with different ion EOS models is recommended if problems occur with two-temperature calculations.

2.3.1 Ideal Gas Model

The ideal gas law is a simple set of relationships describing the properties of a perfect monatomic gas.

$$p_i(\rho, T) = \frac{RT\rho}{M} \quad (2.4)$$

$$u_i(\rho, T) = \frac{3RT}{2M} \quad (2.5)$$

$$a_i(\rho, T) = -\frac{RT}{M} \left(-7.072343 + \frac{3}{2} \ln(MT) + \ln\left(\frac{M}{\rho}\right) \right) \quad (2.6)$$

Equation (2.6) was taken directly from the OpenSesame software [3], which is used to generate SESAME EOS data. Equation (2.1) supplements equations (2.4) through (2.6) to calculate the entropy data.

Curiosity drives the author to determine the origin of equation (2.6). The entropy differential (T dS equation) is defined as equation (2.7).

$$ds = \frac{du}{T} + \frac{p}{T} dv \quad (2.7)$$

Equation (2.7) may be rewritten as equation (2.8).

$$ds = \frac{du}{dT} \frac{dT}{T} + \frac{R}{M} \frac{dv}{v} \quad (2.8)$$

Given $\frac{du}{dT} = \frac{3R}{2M}$ from equation (2.5), equation (2.8) yields equation (2.9).

$$\int ds = \int \frac{3R}{2M} \frac{dT}{T} + \int \frac{R}{M} \frac{dv}{v} \quad (2.9)$$

Integrating by substitution ($f = MT$, $df = dT$ and $g = Mv$, $dg = dv$) equation (2.9) results in equation (2.10).

¹ Two-temperature EOS data allows a host code to perform calculations with temperature fields associated with ions and electrons separately.

$$s = \frac{3R}{2M} \ln(MT) + \frac{R}{M} \ln(Mv) + s_0 \quad (2.10)$$

The Helmholtz free energy is determined by combining equations (2.1) and (2.10) to yield equation (2.11), where $v \equiv \frac{1}{\rho}$.

$$a = \frac{3RT}{2M} - \frac{RT}{M} \left(s_0 + \frac{3}{2} \ln(MT) + \ln\left(\frac{M}{\rho}\right) \right) \quad (2.11)$$

Equation (2.11) can be rewritten as equation (2.12).

$$a = -\frac{RT}{M} \left(\left(s_0 - \frac{3}{2} \right) + \frac{3}{2} \ln(MT) + \ln\left(\frac{M}{\rho}\right) \right) \quad (2.12)$$

The general form of equations (2.6) and (2.12) are identical. The value of $\left(s_0 - \frac{3}{2} \right)$ is the result of applying the ideal gas limit for a monatomic gas, and it is beyond the scope of this document.

2.3.2 Cowan Model

This section describes the simple analytical model developed by R. D. Cowan and documented for the IONEOS Fast, Analytic, Ion Equation-of-State Routine [4]. A normalized local mass density and a dimensionless constant are defined by equations (2.13) and (2.14) respectively.

$$\xi = \frac{9Z^{0.3}\rho}{M} \quad (2.13)$$

$$\beta = 0.6Z^{1/6} \quad (2.14)$$

It is convenient to define the specific heat relationship:

$$c_v = \left. \frac{\partial u}{\partial T} \right|_v = T \left. \frac{\partial s}{\partial T} \right|_v. \quad (2.15)$$

The Debye temperature and the Lindemann melting temperature are defined by equations (2.16) and (2.17) respectively.

$$T_D = \frac{(1.68)\xi^{2+\beta}}{(Z+22)(1+\xi)^2} \quad (2.16)$$

$$T_M = \frac{(0.32)\xi^{4+2\beta-\frac{2}{3}}}{(1+\xi)^4} \quad (2.17)$$

The ion temperature variables (ϕ) and Gruneisen parameters (γ) are described in equations (2.18) through (2.23) for the fluid (F) and solid (S) phases.

$$\phi_F = \left(\frac{T_M}{T_i} \right)^{\frac{1}{3}} \quad (2.18)$$

$$\phi_S = \frac{T_D}{T_i} \quad (2.19)$$

$$\gamma_F = 3\beta - 1 + \frac{6}{(1+\xi)} \quad (2.20)$$

$$\gamma_S = \beta + \frac{2}{(1+\xi)} \quad (2.21)$$

$$\gamma'_F = \gamma_F + \frac{2\gamma_F^2}{9} + \frac{6\xi}{(1+\xi)^2} \quad (2.22)$$

$$\gamma'_S = \beta + \frac{2}{(1+\xi)^2} \quad (2.23)$$

Use equations (2.24) through (2.26) for the fluid region ($T_i > T_M$).

$$p_i = \frac{RT\rho}{M}(1 + \gamma_F\phi_F) \quad (2.24)$$

$$u_i = \frac{3RT}{2M}(1 + \phi_F) \quad (2.25)$$

$$s_i = \frac{R}{M} \left[7 - 3\phi_F + \frac{3}{2} \ln \left(\frac{0.02T_i}{\left(\frac{0.42}{22+Z} \right)^2} \right) - \ln(\xi) \right] \quad (2.26)$$

Equation (2.26) was taken directly from the OpenSesame software [3], and it can be shown to satisfy the specific heat relation of equation (2.15). Equation (2.1) supplements equations (2.24) through (2.26) to calculate the Helmholtz free energy data.

Use equations (2.27) through (2.29) for the high-temperature solid region ($T_i \leq T_M$ and $3T_i > T_D$).

$$p_i = \rho \gamma_s u_i \quad (2.27)$$

$$u_i = \frac{3RT}{M} \left(1 + \frac{\phi_s^2}{20} - \frac{\phi_s^4}{1680} \right) \quad (2.28)$$

$$s_i = \frac{R}{M} \left[4 + 3 \left(\phi_s^2 \left(\frac{1}{40} - \frac{\phi_s^2}{2240} \right) - \ln(\phi_s) \right) \right] \quad (2.29)$$

Equation (2.29) was taken directly from the OpenSesame software [3] , and it can be shown to satisfy the specific heat relation of equation (2.15). Equation (2.1) supplements equations (2.27) through (2.29) to calculate the Helmholtz free energy data.

Use equations (2.30) through (2.32) for the low-temperature solid region ($T_i \leq T_M$ and $3T_i \leq T_D$).

$$p_i = \rho \phi_s u_i \quad (2.30)$$

$$u_i = \frac{3RT}{M} \left(\frac{3}{8} \phi_s + \frac{\pi^4}{5\phi_s^3} - \left(3 + \frac{9}{\phi_s} + \frac{18}{\phi_s^2} + \frac{18}{\phi_s^3} \right) e^{-\phi_s} \right) \quad (2.31)$$

$$s_i = \frac{R}{M} \left[4 \left(\frac{\pi^4}{5\phi_s^3} - \left(\frac{9}{4} + \frac{9}{\phi_s} + \frac{18}{\phi_s^2} + \frac{18}{\phi_s^3} \right) e^{-\phi_s} \right) \right] \quad (2.32)$$

Equation (2.32) was analytically derived using equations (2.2) and (2.3). Equation (2.1) supplements equations (2.30) through (2.32) to calculate the Helmholtz free energy data.

It is important to note that the Cowan Model may introduce unwanted pathologies due the fact that its functions are discontinuous at $\phi_s = 3$. Figure 1 demonstrates the aforementioned discontinuity between equations (2.29) and (2.32), and it is quantified to be approximately a three percent deviation.

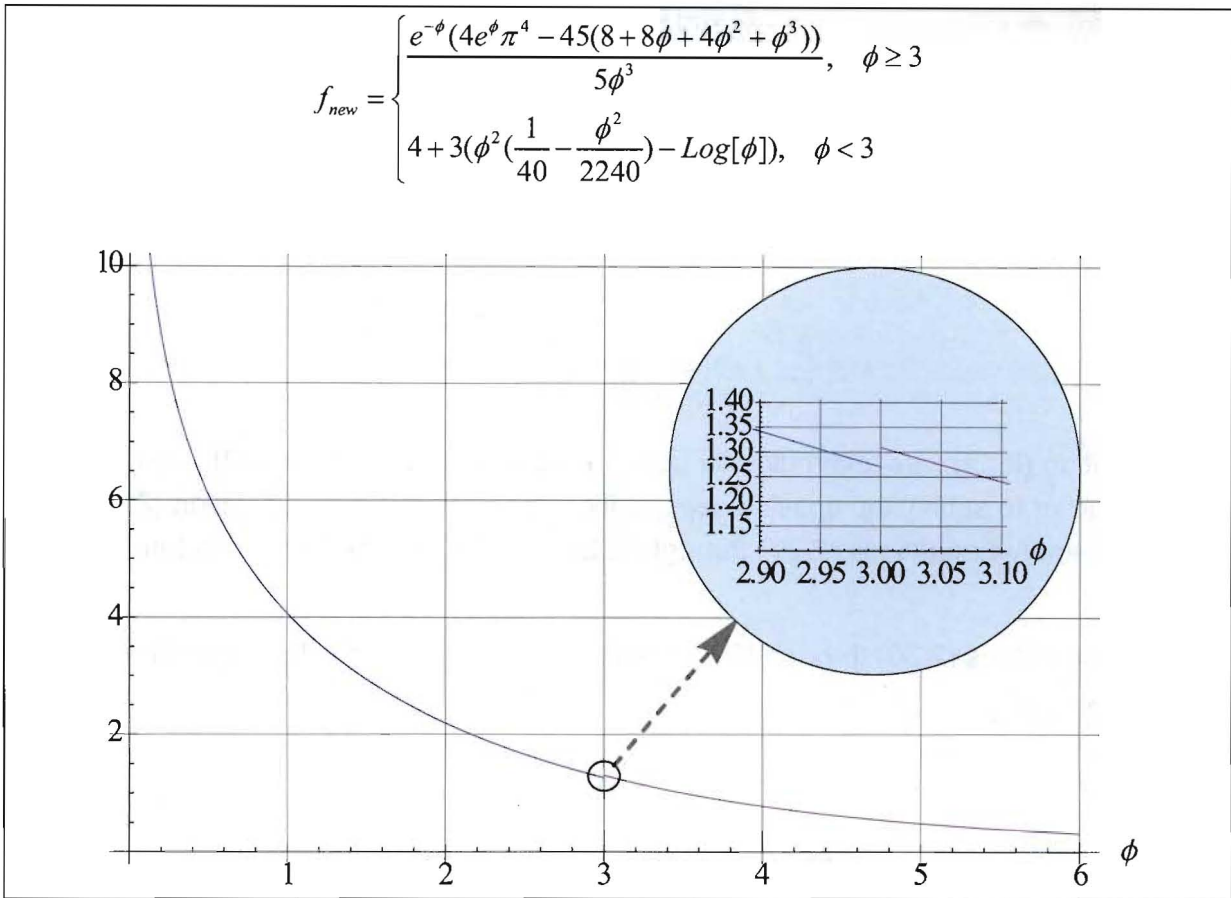


Figure 1. Dimensionless parameters from the high-temperature solid entropy expression and the new low-temperature solid entropy expression.

2.3.3 Number Proportional Model

Since a subtraction of the analytical model values from the tabulated total EOS data is performed to calculate an electron EOS, pathologies will typically exist within the resultant data at low temperatures and high densities due to the fact that the chosen ion EOS was not used to calculate the original EOS data. The number proportional model, in principle, albeit not always, mitigates such pathological data in that it uses simple ratio equations to model the ion EOS [2].

$$p_i(\rho, T) = \frac{p(\rho, T)}{1 + Z} \tag{2.33}$$

$$u_i(\rho, T) = \frac{u(\rho, T)}{1 + Z} \tag{2.34}$$

$$a_i(\rho, T) = \frac{a(\rho, T)}{1 + Z} \quad (2.35)$$

Equations (2.1) through (2.3) are used to calculate the entropy and, subsequently, the Helmholtz free energy in the event that no Helmholtz free energy data is tabulated; otherwise, equation (2.1) supplements equations (2.33) through (2.35) to calculate the entropy data.

The number of free electrons per ion is estimated by assuming the thermal electron EOS is determined using the Fermi-gas model.

$$Z = Z_1 F_{1/2}(\eta) \quad (2.36)$$

$$Z_1 = \frac{2}{N_i} \left(\frac{mkT}{2\pi\hbar^2} \right)^{3/2} \quad (2.37)$$

$$\frac{Z_0 Z}{1 + Z} = Z_1 F_{3/2}(\eta) \quad (2.38)$$

$$Z_0 = \frac{p(\rho, T) - p_c(\rho)}{N_i kT} \quad (2.39)$$

The Fermi integrals satisfy equation (2.40) to at least one-percent accuracy.

$$F_{3/2} = F_{1/2} \left(1 + (0.88388) F_{1/2} + (0.37208) F_{1/2}^2 + (0.02645) F_{1/2}^{10/3} \right)^{1/5} \quad (2.40)$$

Upon substituting equations (2.36) and (2.38) into (2.40), equation (2.41) is produced.

$$Z + 1 = Z_0 \left(1 + (0.88388) \left(\frac{Z}{Z_1} \right) + (0.37208) \left(\frac{Z}{Z_1} \right)^2 + (0.02645) \left(\frac{Z}{Z_1} \right)^{10/3} \right)^{1/5} \quad (2.41)$$

Equation (2.41) can be solved iteratively, and it is constrained by equations (2.42) and (2.43).

$$Z \geq 0 \quad (2.42)$$

$$Z_0 \geq 1 \quad (2.43)$$

2.4 Additional Thermodynamic Quantities

Often users of EOSPAC are interested in calculating quantities, which are not directly provided by the EOSPAC interface. Distributed with EOSPAC is a utility

named `get_sesame_data`, which provides a command line interface to various EOSPAC capabilities like querying the content of SESAME data file(s). Additionally, `get_sesame_data` can calculate various derived thermodynamic values, which are described in this section.

Given density (ρ) and temperature (T), calculate the following: pressure (p), specific internal energy (u), specific Helmholtz free energy (a), specific entropy (s), sound speed (c), adiabatic bulk modulus (β), Gruneisen Coefficient (Γ), inverse isothermal compressibility (ρc_T^2), and specific heats (c_v and c_p). The pressure, specific internal energy, specific Helmholtz free energy, and specific entropy are simply calculated by interpolating the respective SESAME data at the given density and temperature. The other quantities require more effort as described in the following sections.

2.4.1 Identities

$$\left. \frac{\partial y}{\partial z} \right|_x \left. \frac{\partial z}{\partial x} \right|_y \left. \frac{\partial x}{\partial y} \right|_z = -1 \quad (2.44)$$

$$\left. \frac{\partial y}{\partial x} \right|_z \left. \frac{\partial x}{\partial y} \right|_z = 1 \quad (2.45)$$

$$\frac{\partial f}{\partial v} = \rho^2 \frac{\partial f}{\partial \rho} \quad \text{where } \rho \equiv \frac{1}{v} \quad (2.46)$$

2.4.2 Sound speed

The sound speed is defined by equation (2.47).

$$c^2 = -v^2 \left. \frac{\partial p}{\partial v} \right|_s = \left. \frac{\partial p}{\partial \rho} \right|_s \quad (2.47)$$

Using equation (2.44), equation (2.47) can be rewritten as equation (2.48).

$$c^2 = \frac{\left. \frac{\partial s}{\partial \rho} \right|_p}{\left. \frac{\partial s}{\partial p} \right|_\rho} \quad (2.48)$$

Equation (2.48) is a simple means to validate the sound speed calculation.

2.4.3 Isentropic Compressibility

The adiabatic bulk modulus is defined by equation (2.49).

$$\beta = \rho c^2 \quad (2.49)$$

The isentropic compressibility is subsequently defined by equation (2.50).

$$K_s = \frac{1}{\beta} = \left(\rho \frac{\partial p}{\partial \rho} \Big|_s \right)^{-1} = - \frac{1}{v} \frac{\partial v}{\partial p} \Big|_s \quad (2.50)$$

2.4.4 Isothermal Compressibility

This isothermal compressibility is defined by equation (2.51).

$$K_T = \frac{1}{\rho c_T^2} = \left(\rho \frac{\partial p}{\partial \rho} \Big|_T \right)^{-1} = - \frac{1}{v} \frac{\partial v}{\partial p} \Big|_T \quad (2.51)$$

The $\frac{\partial p}{\partial \rho} \Big|_T$ partial derivative is a calculated side effect of interpolating the tabulated data for $p = p(\rho, T)$.

It is of interest to verify the constraint of equation (2.52).

$$c_T^2 \leq c^2 \quad (2.52)$$

2.4.5 Gruneisen Coefficient

The Gruneisen Coefficient is defined by equation (2.53).

$$\Gamma = \frac{1}{\rho} \frac{\partial p}{\partial u} \Big|_\rho \quad (2.53)$$

The $\frac{\partial p}{\partial u} \Big|_\rho$ partial derivative is a calculated side effect of interpolating the tabulated data for $p = p(\rho, u)$.

2.4.6 Specific heats

The constant volume specific heat is defined by equation (2.54).

$$c_v = \frac{\partial u}{\partial T} \Big|_v = \frac{\partial u}{\partial T} \Big|_\rho \quad (2.54)$$

The $\left. \frac{\partial u}{\partial T} \right|_p$ partial derivative is a calculated side effect of interpolating the tabulated data for $u = u(\rho, T)$.

The constant pressure specific heat is defined by equation (2.55).

$$c_p = T \left. \frac{\partial s}{\partial T} \right|_p = \left. \frac{\partial h}{\partial T} \right|_p \quad (2.55)$$

Unfortunately, at this point, we find that the constant pressure specific heat cannot be calculated using EOSPAC 6's interpolation results. This is due to the fact that the $\left. \frac{\partial s}{\partial T} \right|_p$ is not available since EOSPAC 6 does not calculate $s = s(p, T)$. In an

attempt to derive an alternative equation, the following derivation is performed. The specific enthalpy is defined by equation (2.56).

$$h = u + pv \quad (2.56)$$

Using equations (2.55) and (2.56), equation (2.57) is derived.

$$c_p = \left. \frac{\partial u}{\partial T} \right|_p + v \left. \frac{\partial p}{\partial T} \right|_v + p \left. \frac{\partial v}{\partial T} \right|_p = \left. \frac{\partial u}{\partial T} \right|_p + p \left. \frac{\partial v}{\partial T} \right|_p \quad (2.57)$$

Given equation (2.44), the $\left. \frac{\partial v}{\partial T} \right|_p$ partial derivative is alternatively defined by equation (2.58).

$$\left. \frac{\partial v}{\partial T} \right|_p = - \frac{\left. \frac{\partial p}{\partial T} \right|_v}{\left. \frac{\partial p}{\partial v} \right|_T} \quad (2.58)$$

Using equations (2.46) and (2.58), equation (2.57) can be rewritten as equation (2.59).

$$c_p = \left. \frac{\partial u}{\partial T} \right|_p + \frac{p}{\rho^2} \left. \frac{\partial p}{\partial \rho} \right|_T \left. \frac{\partial p}{\partial T} \right|_v \quad (2.59)$$

Unfortunately, the $\left. \frac{\partial u}{\partial T} \right|_p$ partial derivative is generally-unavailable using EOSPAC

6's interpolation methods on the SESAME data; therefore, an alternative form is required. Consider the ratio of specific heats as defined in equation (2.60).

$$\kappa = \frac{c_p}{c_v} \quad (2.60)$$

Equation (2.60) can be rewritten as equation (2.61).

$$\kappa = \frac{T \left. \frac{\partial s}{\partial T} \right|_v}{T \left. \frac{\partial s}{\partial T} \right|_p} \quad (2.61)$$

Using equation (2.44), equation (2.61) can be rewritten as equation .

$$\kappa = \frac{\left. \frac{\partial v}{\partial s} \right|_T \left. \frac{\partial T}{\partial v} \right|_s}{\left. \frac{\partial p}{\partial s} \right|_T \left. \frac{\partial T}{\partial p} \right|_s} = \left(\left. \frac{\partial v}{\partial s} \right|_T \left. \frac{\partial s}{\partial p} \right|_T \right) \left(\left. \frac{\partial T}{\partial v} \right|_s \left. \frac{\partial p}{\partial T} \right|_s \right) \quad (2.62)$$

Using the Chain Rule, equations (2.63) and (2.64) are derived.

$$\left. \frac{\partial v}{\partial p} \right|_T = \left. \frac{\partial v}{\partial s} \right|_T \left. \frac{\partial s}{\partial p} \right|_T \quad (2.63)$$

$$\left. \frac{\partial p}{\partial v} \right|_s = \left. \frac{\partial p}{\partial T} \right|_s \left. \frac{\partial T}{\partial v} \right|_s \quad (2.64)$$

Applying equations (2.63) and (2.64) to equation (2.62) yields equation (2.65).

$$\kappa = \left. \frac{\partial v}{\partial p} \right|_T \left. \frac{\partial p}{\partial v} \right|_s \quad (2.65)$$

From equations (2.45), (2.50) and (2.51), equation (2.65) can be rewritten as equation (2.66).

$$c_p = \frac{K_s}{K_T} c_v = \frac{c_T^2}{c^2} c_v \quad (2.66)$$

3 GENERAL INTERFACE DESCRIPTION

This section describes, in general, the EOSPAC interface library and how a host code will use it. Figure 2 shows how the EOSPAC public interface will interact with host codes written in various languages.

Four host code languages are specifically targeted by the public interface of EOSPAC: C++, C, FORTRAN 77, and Fortran 90. As shown in Figure 2, EOSPAC provides a flat² public interface with unmangled³ procedure definitions. This has the distinct advantage of providing the user with consistent data types and procedure interfaces regardless of the host code's language and working platform.

To ensure language interoperability and platform portability EOSPAC Version 6 is written using the POSIX [5, 6] subset of C.

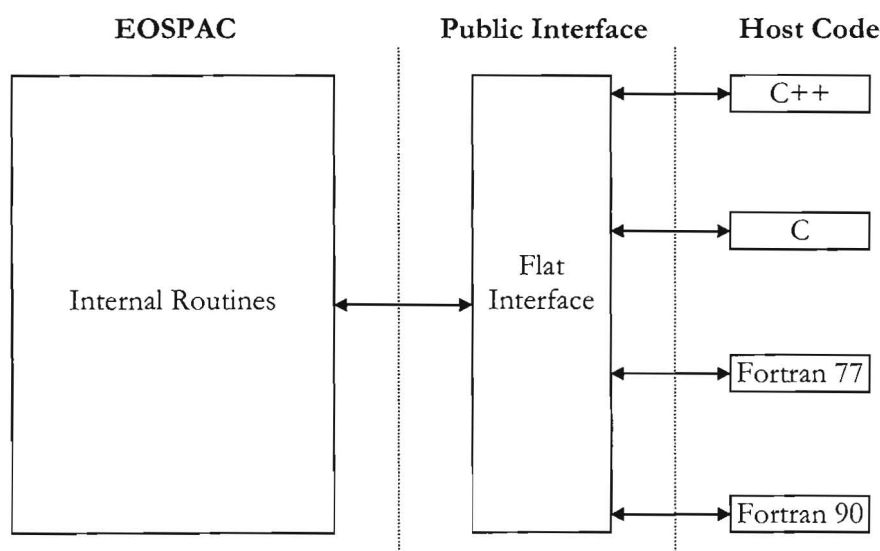


Figure 2. General graphical description of the public user interface of EOSPAC.

² Procedure arguments are reduced to a set of basic data types common to all applicable programming languages.

³ Procedure names are ensured to be visible, unique and sensible across the multiple-programming-language interface. In software compilation, name mangling (sometimes called name decoration) is a technique used to solve various problems caused by the need to resolve unique names for programming entities in many modern programming languages.

3.1 Usage Examples

The usage examples give an overview of typical user interactions with EOSPAC. There are only two such examples, a serial host code case and a parallel host code case.

3.1.1 Serial Case

The serial case is shown in Figure 3. During the host code's setup phase the data tables are loaded, and setup options may be set or reset prior to and/or after the data is actually loaded into memory. During the host code's calculation phase the data of selected tables is accessed using either interpolation or mixing, and interpolation/mixing options may be set or reset prior to and/or after the data is actually accessed. This is done N times where N is problem dependent, but should include at least once per data table. An optional step is to get information and comments about the data tables, for example for debugging purposes. This is done M times where M can vary from zero to the number of data tables. The data tables are destroyed when the user is done with them.

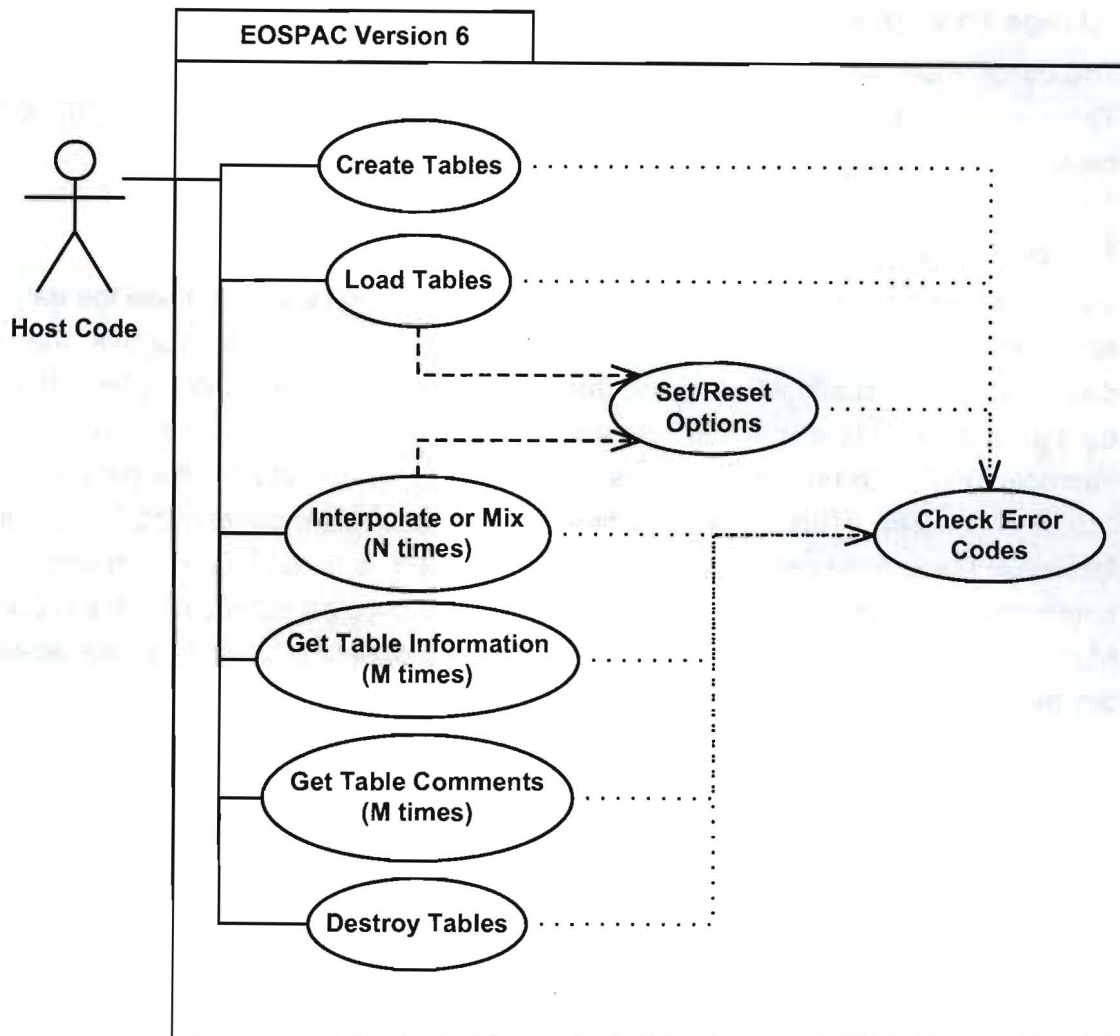


Figure 3. Serial host code use of EOSPAC.

3.1.2 Parallel Case

The parallel case is shown in Figure 4. The “Load Tables” occurs on a single process and is identical to the serial case. The same process then gets the size of the packed tables and allocates storage to hold them. The process then extracts the packed tables from EOSPAC. The packed tables are then distributed to all child processes. Each child process then loads its packed tables into EOSPAC. The data is then accessed on each process just as if it was a serial run. Each process then destroys its data tables.

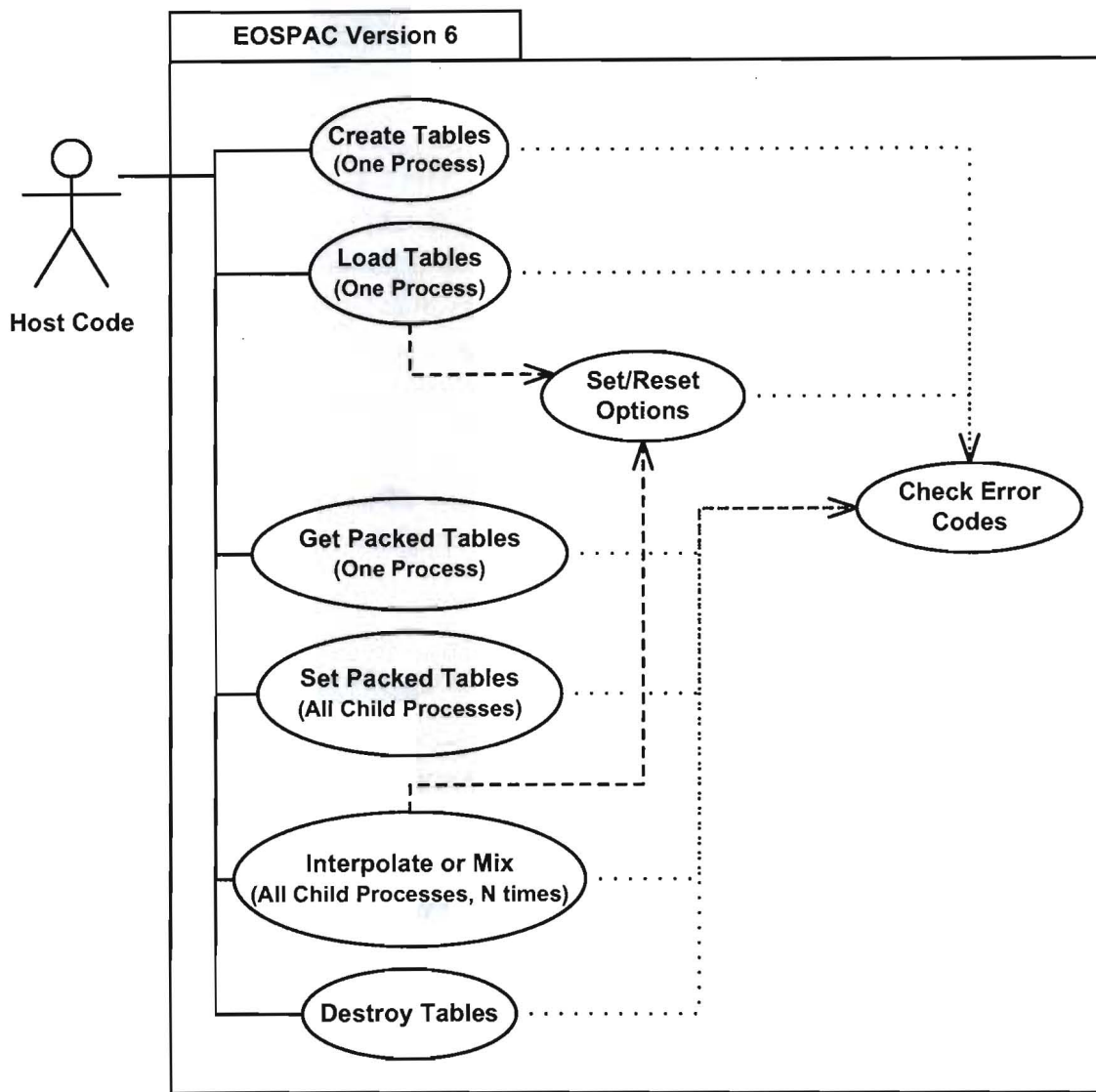


Figure 4. Parallel host code use of EOSPAC.

4 CONVENTIONS

Several conventions are used throughout this document.

4.1 Data Organization

Conceptually EOSPAC is organized around data tables. A data table is specified by the material identification number, by the table type (e.g. pressure as a function of density and temperature), and the processing options (e.g. smoothed, monotonic, etc.). Two data tables differ if any option differs; thus, a smoothed data table is

different than a monotonic data table. This is just common sense because the values returned for the two data tables will be different. The i -th data table will be referred to as T_i .

A table handle is used to access the data table. The table handle is a language independent mechanism for a host code to access a specific instance of the data tables being managed by EOSPAC. Note that table handles are not implemented using native language pointers. The details of establishing a table handle is discussed in section 5 and usage is shown in sections 6 through 7.

Multiple table handles are returned from the setup routine within a user-supplied array. The host code then uses the table handles to specify on which data tables EOSPAC is to operate. Typical operations are interpolating to get data at points desired by the host code, and to destroy the data tables.

4.2 Routine Names

Routine name standardization is applied according to the following rules:

1. EOSPAC is a package of routines that provides a cohesive set of logically related functionality to host codes. The package name "eos_" is used as a prefix for all routine names in the package. This practically guarantees unique routine names when linked to the host codes. The prefix of a routine name allows users to instantly identify the physical package from which it came, and the prefix gives users a hint about functionality.
2. A routine name takes the form of ActionSubset where Action specifies a given operation and the optional Subset specifies a property, information, etc. The complete name will be eos_ActionSubset.
3. The names of certain actions on tables have been standardized. The standardized action names are as follows:
 - "Create" will instantiate data object(s) to store a table or collection of tables
 - "Destroy" will destroy a table or collection of tables
 - "Get" retrieves information about a table
 - "Interpolate" performs interpolation using the table's member data
 - "Load" will create a new table and fill the table's members with appropriate information
 - "Reset" reasserts any default information to a table (i.e., option setting)

- "Set" assigns information to a table (i.e., option setting)

To summarize, routine names are generally defined by eos_ActionSubset.

4.3 Constant Identifier Names

Names of constant identifiers available to host codes are standardized by applying the following rules:

1. All identifiers begin with the following four characters: "EOS_".
2. The underscore is used to separate words if the name is comprised of multiple words.

All EOSPAC routine names and identifiers are limited to 31 characters. This is done to avoid exceeding the limit on some compilation systems.

4.4 Data Types

Throughout this document language data types will be referred to generically. The actual definition is machine, language, and compiler specific. The data types used by EOSPAC are:

- EOS_INTEGER a 32-bit signed integer data type
- EOS_REAL a 64-bit signed floating point data type
- EOS_CHAR an 8-bit character type.

Some parameters of data type EOS_INTEGER that are related to the data types are:

- EOS_TRUE a constant specifying a Boolean true
- EOS_FALSE a constant specifying a Boolean false
- EOS_MaxErrMsgLen a constant specifying the maximum character string length associated with an EOSPAC error message

5 SETUP MATERIAL DATA

The setup phase consists of calls to an interface routine that establishes EOSPAC data tables and loads them with appropriate data. In addition to this setup routine, there exist routines to destroy data tables, pack their member data into a portable array, and

unpack such an array into data tables. The packed array features allow multithreaded host codes to share data between processes if necessary.

5.1 Data Locations

Before any description of how data is loaded, discarded, packed or unpacked within memory, it is vital to know how EOSPAC is able to find the SESAME data files desired. To do so, three algorithms are used to build a list of file names: 1) Environment-variable-defined and default search paths, 2) Index file, and 3) Default file name list. Once all of these algorithms are completed, the result is an ordered list of absolute-referenced file names that is subsequently edited to remove all duplicate file references. File attributes and, if necessary, bitwise file comparisons are made to eliminate any duplication of files. It is important to note here that two files are not considered duplicates if only part of the contained data is identical.

The ordered list of file names is written to the TablesLoaded.dat file when either the EOS_APPEND_DATA or EOS_DUMP_DATA option is set (see APPENDIX D).

5.1.1 Environment-variable-defined and default search paths

Initially, the current working directory is put at the top of an ordered list of search paths. If EOSPAC detects that the current environment has set the variable named SESAMEPATH, it parses it for a list of search paths. Within the UNIX and Windows environments, this environment variable is delimited by colons and semicolons respectively. These path names are appended to the ordered list of paths. Finally, a default list of search paths is appended to the ordered list of paths:

Description	Path Name
LANL Production data path	/usr/projects/data/eos
LANL X-Div LAN data path	/usr/local/codes/data/eos
LANL Cray unclassified data path	/usr/local/udata/ses
LANL Cray classified data path	/usr/local/cdata
LLNL Production data path	/usr/gapps/lanl-data/eos
SANDIA Production data path	/projects/lanl-data/eos

5.1.2 Ordered File Names List Creation

For each of the search paths found by the algorithm described in section 5.1.1, the two remaining algorithms are executed in order. These two remaining algorithms are described in sections 5.1.2.1 and 5.1.2.2 respectively, and Figure 5 contains a flowchart description of how they are implemented.

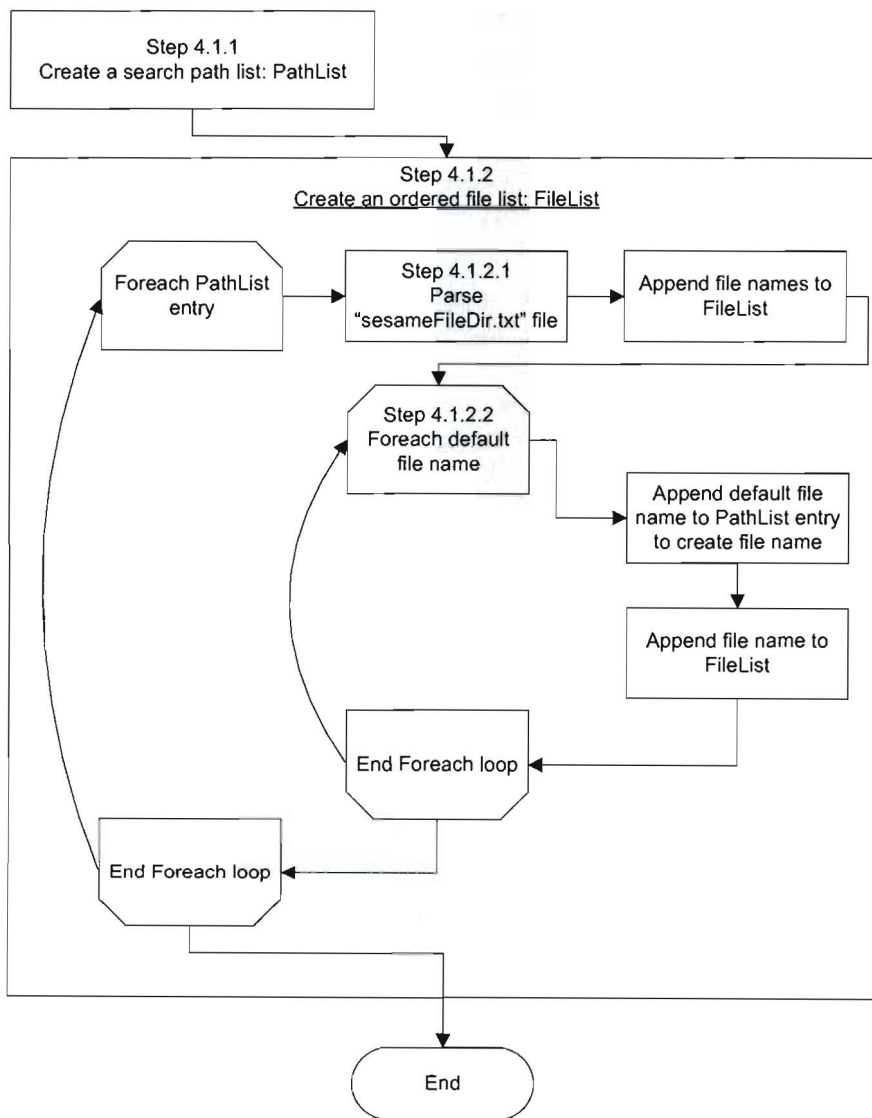


Figure 5. Flowchart description of file search algorithms.

NEW for 6.2.2

As of version 6.2.2, EOSPAC will parse a “sesameFilesDir.txt” found in the current working directory every time the eos_CreateTables routine is called. This modification allows the host code to dynamically incorporate changes to the ordered files list.

5.1.2.1 Index file

EOSPAC tests for the existence of an index file, a text file named “sesameFilesDir.txt” (Figure 6), within the specified search path found by the algorithm described in section 5.1.1.

```
# Distributed Sesame file list

# Unix absolute reference (two file names per line)
/usr/local/codes/data/eos/sesame;/usr/local/codes/data/eos/sescu

# Unix absolute reference (one file name per line)
/usr/local/codes/data/eos/sescul
/usr/local/codes/data/eos/sescu9
/usr/local/codes/data/eos/sesou

# DOS/Windows absolute reference (two file names per line)
I:\data\eos\sesame;I:\data\eos\sescu

# alternative DOS/Windows absolute references
\\xfiles\codes\data\eos\sescul
\\xfiles\codes\data\eos\sescu9
\\xfiles\codes\data\eos\sesou

# relative references with respect to this index file's location
export-controlled/ieee64/sesame;export-controlled/ieee64/sescu
export-controlled/ieee64/sescul
export-controlled/ieee64/sescu9
export-controlled/ieee64/sesou

# associate material id and Sesame file
MATID 9001 sesame3
MATID 9002 ../../../../sesame3
MATID 9003 /usr/local/codes/data/eos/sesame3
```

Figure 6. Example of sesameFilesDir.txt.

If the index file is found, it is parsed according to the following rules to find references to SESAME data files:

- Delimiters include linefeed, carriage return, and semicolon.
- Comments are ignored and begin with #.
- Leading white space is ignored.
- Paths that are relative to the opened index file are converted to absolute paths.
- Invalid file names are silently ignored. A file name is invalid if it doesn't exist or is exceeds the maximum number of characters (PATH_MAX) for the current file system. The value of PATH_MAX is discussed further in section 5.3.4.
- **NEW for 6.2.1** If the case-sensitive token, END, is found as the first non-whitespace characters on a line in the index file, then no other files will be added to the ordered file list, which is defined in section 5.1.2. This feature is available as of version 6.2.1.
- **NEW for 6.2.2** If the case-sensitive token, MATID, is found as the first non-whitespace characters on a line in the index file, then the remainder of the line shall contain a material ID (integer) and the associated SESAME file name. A file association to a material ID supercedes any previous associations (e.g., associating 9001 to sesame3 and then to sesame2 will retain the last association). See Figure 6 for examples. *It is important to note that once a material ID is associated with a specific SESAME file, the association will remain until either the code terminates or the another explicit association is provided – there exists no mechanism to reset to the default data search algorithm.* This feature is available as of version 6.2.2.
- It is important to note that the MATID and END tokens constrain the data loaded for all table handles (i.e., it is a global effect). To set table handle-specific constraints, see the eos_GetMaxDataFileNameLength and eos_SetDataFileName functions described in sections 5.3.4 and 5.3.8 respectively.

Once parsed, the list of file names found in "sesameFilesDir.txt" is appended to the list of SESAME data file names to be searched.

5.1.2.2 Default file name list

For compatibility with earlier versions of EOSPAC and old distributions of SESAME files, a default list of filenames has been preserved. This ordered list of filenames is provided in Table 1. This list of file names, if found within the specified search path found by the algorithm described in section 5.1.1, is appended to the ordered list of files that will be searched for any requested data.

Table 1. Ordered list of default SESAME file names.

File Name	File Name	File Name
1 sesameu	2 Sesameu1	3 sesameu2
4 sesameu3	5 sesameu4	6 sesamea
7 sesamea1	8 sesamea2	9 sesameb
10 sesamec	11 sesame	12 sesame1
13 sesame2	14 sesame3	15 sesame4
16 sesep	17 sesep1	18 sesep2
19 sesep3	20 sesep4	21 sesou
22 sesou1	23 sesou2	24 sesou3
25 sesou4	26 sesop	27 sesop1
28 sesop2	29 sesop3	30 sesop4
31 sescu	32 sescu1	33 sescu2
34 sescu3	35 sescu4	36 sescu9
37 sescp	38 sescp1	39 sescp2
40 sescp3	41 sescp4	

5.1.3 Ordered File Names List Example

Assume that the current working directory is defined as follows:

```
~/FILES/eospac6.00branch/Source/tests
```

Assume that the following SESAME data files exist for the machine being used:

```
~/FILES/tmp/tests/data/sesame1
~/FILES/tmp/tests/data/sesame3
~/FILES/sesame/081105/sesame_bin_081105
~/FILES/sesame/081105/sesame_bin_081105_sgi
~/FILES/sesame/081105/sesame
~/FILES/code/b11/test/sesame/sesame
~/FILES/code/b11/test/sesame/sescresu
~/FILES/code/b11/test/sesame/sescu
~/FILES/code/b11/test/sesame/sescu1
```

EOSPAC User's Manual: Version 6.2

```
~/FILES/code/bll/test/sesame/sescu9
~/FILES/code/bll/test/sesame/sesou
~/FILES/eospac6_mainbranch/Source/tests/data/sesame1
~/FILES/eospac6_mainbranch/Source/tests/data/sesame3
~/FILES/eospac6_mainbranch/Source/tests/alt_tests/sesameFilesDir.txt
~/FILES/eospac6_mainbranch/Source/tests/alt_tests/lambda_parallel/sesame3
~/FILES/eospac6_automake_tests/Source/tests/data/sesame1
~/FILES/eospac6_automake_tests/Source/tests/data/sesame3
~/FILES/eospac6.00branch/Source/tests/data/sesame1
~/FILES/eospac6.00branch/Source/tests/data/sesame3
~/FILES/eospac6.00branch/Source/tests/sesameFilesDir.txt
~/FILES/eospac6.10alpha.7/Source/tests/data/sesame1
~/FILES/eospac6.10alpha.7/Source/tests/data/sesame3
~/FILES/eospac6.10alpha.7/Source/tests/sesameFilesDir.txt
~/FILES/eospac6.10alpha.7/Source/tests/alt_tests/sesameFilesDir.txt
~/FILES/eospac6.10alpha.7/Source/tests/alt_tests/lambda_parallel/sesame3
```

Assume that “sesameFilesDir.txt” in the current working directory that contains the following information:

```
# Sesame3 test data file list
./data/sesame3

# Sesame1 test data file list
./data/sesame1
```

Assume the value of the SESAMEPATH environment variable to contain

```
“/usr/projects/data/eos/export-controlled/ieee64:${HOME}/FILES
/eospac6.10alpha.7/Source/tests/data:${HOME}/FILES/code/bll/t
est/sesame”
```

Given all of the above assumptions, the ordered list of files names would be as follows:

0. ../data/sesame3
1. ../data/sesame1
2. /usr/projects/data/eos/export-controlled/ieee64/sesame
3. /usr/projects/data/eos/export-controlled/ieee64/sesou
4. /usr/projects/data/eos/export-controlled/ieee64/sescu
5. /usr/projects/data/eos/export-controlled/ieee64/sescul
6. /usr/projects/data/eos/export-controlled/ieee64/sescu9
7. /users/myhome/./FILES/eospac6.10alpha.7/Source/tests/data/sesame1
8. /users/myhome/./FILES/eospac6.10alpha.7/Source/tests/data/sesame3
9. /users/myhome/FILES/code/bll/test/sesame/sesame
10. /users/myhome/FILES/code/bll/test/sesame/sesou
11. /users/myhome/FILES/code/bll/test/sesame/sescu
12. /users/myhome/FILES/code/bll/test/sesame/sescul
13. /users/myhome/FILES/code/bll/test/sesame/sescu9
14. /usr/projects/data/eos/sesame

- 15. /usr/projects/data/eos/sesou
- 16. /usr/projects/data/eos/sescu
- 17. /usr/projects/data/eos/sescul
- 18. /usr/projects/data/eos/sescu9

5.2 Data Organization

As briefly described in the section introduction above, the loaded data is referenced by unique table handles. The arguments of the interface routines are organized into a set of ordered arrays such that each array element corresponds to a data table.

For example, the table types (see APPENDIX B and APPENDIX C), SESAME material ID numbers, table options (see APPENDIX A) and error codes (see APPENDIX G) are stored within identically dimensioned arrays (see Figure 7). Each row of Figure 7 specifies a data table that is referenced by the table handle. This conceptual organization is used for all the setup routine arguments that are arrays.

<u>Table Handle</u>	<u>Table Type</u>	<u>Material ID</u>	<u>Table Options</u>	<u>Error Codes</u>
tableHandle ₁	tableType ₁	matID ₁	tableOptions ₁	errorCode ₁
tableHandle ₂	tableType ₂	matID ₂	tableOptions ₂	errorCode ₂
.
.
.
tableHandle _N	tableType _N	matID _N	tableOptions _N	errorCode _N

Figure 7. Input/output data organization.

If the host specifies the loading of identical data for multiple table handles (inadvertently or otherwise), then EOSPAC will make the two table handles identical (Figure 8). This practice is not recommended because it unnecessarily complicates the loaded data's organization.

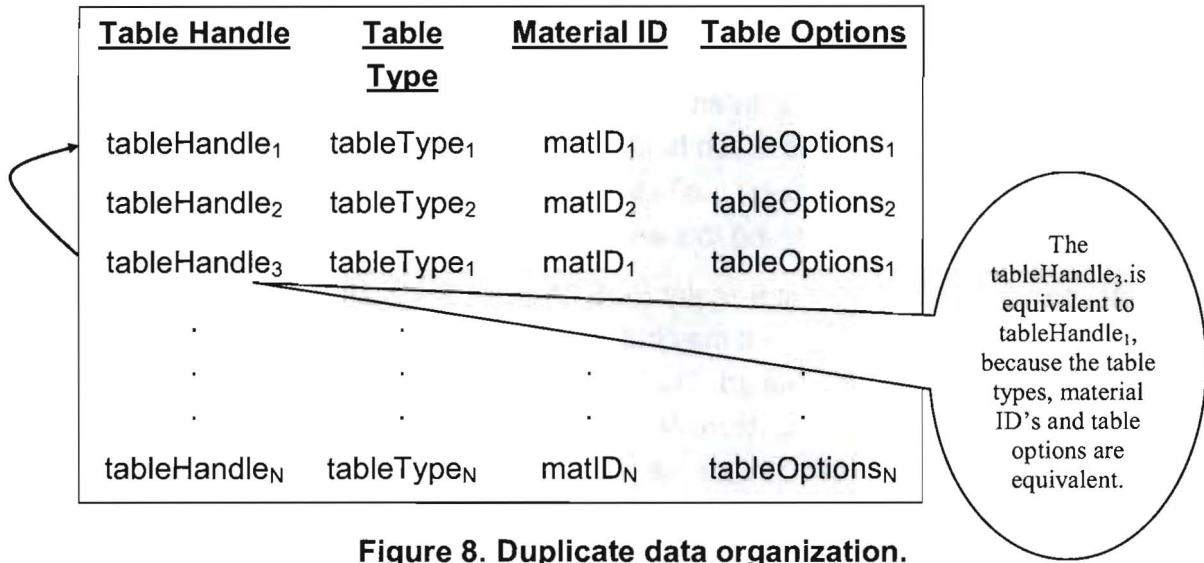


Figure 8. Duplicate data organization.

5.3 Routines and Parameters

The routines and parameters for setting up the material data are discussed in this section.

5.3.1 eos_CreateTables

The eos_CreateTables routine allocates all memory to store the specified data tables. After calling eos_CreateTables, the host code may need to call eos_SetOption so the desired set up options can be changed from the documented defaults.

The input argument is:

- nTables This is the scalar EOS_INTEGER total number of data tables on which to operate.
- tableType This is an EOS_INTEGER array containing the list of table types corresponding to each member data table, T_i, where i=1...nTables. See APPENDIX B and APPENDIX C for table type details.
- matID This is an EOS_INTEGER array containing the SESAME material identification numbers corresponding to each member data table, T_i, where i=1...nTables.

The output arguments are:

tableHandles	This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i=1\dots nTables$. The host code is responsible for managing this array of table handles.
errorCode	This is a scalar EOS_INTEGER variable to contain an error code that may indicate one or more of the tables could not be created. The host code must call eos_GetErrorCode and eos_GetErrorMessage to retrieve error details for a specified tableHandle. See APPENDIX G for error code details.

5.3.2 eos_DestroyAll

The eos_DestroyAll routine releases all memory associated with any remaining data tables and temporary cached data used by EOSPAC routines internally. It is strongly recommended that this routine be used when the currently defined set of SESAME data files is no longer used (i.e., just prior to the end of the host code's execution).

There are no input arguments.

The output arguments are:

errorCode	This is a scalar EOS_INTEGER variable to contain an error code that may indicate failure to release all memory associated with temporary cached data. The host code must call eos_GetErrorMessage to retrieve error details. See APPENDIX G for error code details.
-----------	---

5.3.3 eos_DestroyTables

The eos_DestroyTables routine releases all memory associated with the specified data tables.

The input arguments are:

nTables	This is the scalar EOS_INTEGER total number of data tables on which to operate.
---------	---

`tableHandles` This is an array of `EOS_INTEGER` handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i=1\dots n$ Tables. The host code is responsible for managing this array of table handles.

The output argument is:

`errorCode` This is a scalar `EOS_INTEGER` variable to contain an error code that may indicate one or more of the tables could not be destroyed. The host code must call `eos_GetErrorCode` and `eos_GetErrorMessage` to retrieve error details for a specified `tableHandle`. See APPENDIX G for error code details.

5.3.4 `eos_GetMaxDataFileNameLength`

NEW for 6.2.2 The `eos_GetMaxDataFileNameLength` routine is used to query the maximum number of characters (`PATH_MAX`) for the current file system. This is a file system-dependent value with typical values like those shown in Table 2.

Table 2. Some typical values of `PATH_MAX`.

File System	Length (bytes)
Mac OSX (i386 and PPC)	1024
Modern Linux (i686 and x86_64)	4096
Solaris (Sparc)	1024
Windows/Cygwin	260

The output argument is

`max_length` This is a scalar `EOS_INTEGER` to contain the definition of `PATH_MAX`.

5.3.5 eos_GetPackedTables

The eos_GetPackedTables routine fills a character array with the specified data table's data. The eos_GetPackedTables routine is used to extract the data tables from EOSPAC to allow multithreaded codes to share the data. This routine is also useful for preparing data tables to be written to a host code's binary restart file.

Before calling this routine the host code must call eos_GetPackedTablesSize to determine packedTablesSize, the total number of bytes required to contain the data associated with the specified data tables, allowing the host code to allocate adequate storage.

The input arguments are:

nTables	This is the scalar EOS_INTEGER total number of data tables on which to operate.
tableHandles	This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i=1\dots nTables$. The host code is responsible for managing this array of table handles.

The output arguments are:

packedTables	This is an array of EOS_CHAR that is used to store all of the member data of specified data tables. This array is designed to allow the host code to share data between multiple processors. If dynamic memory allocation for arrays is not possible, then this routine will prove difficult to use since it is to be allocated to hold packedTablesSize characters, where packedTablesSize is returned from the eos_GetPackedTablesSize routine.
errorCode	This is a scalar to contain the error code. See APPENDIX G for error code details.

5.3.6 eos_GetPackedTablesSize

The eos_GetPackedTablesSize routine calculates the total number of bytes required to contain the data associated with the specified data tables. The eos_GetPackedTablesSize routine is used with the eos_GetPackedTables routine.

The input arguments are:

nTables	This is the scalar EOS_INTEGER total number of data tables on which to operate.
tableHandles	This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i=1\dots nTables$. The host code is responsible for managing this array of table handles.

The output arguments are:

packedTablesSize	This is the scalar EOS_INTEGER number of bytes required to hold a specified list of data tables' member data – size of packedTables.
errorCode	This is a scalar EOS_INTEGER variable to contain an error code that may indicate failure to pack the data. The host code must call eos_GetErrorCode and eos_GetErrorMessage to retrieve error details for a specified tableHandle. See APPENDIX G for error code details.

5.3.7 eos_LoadTables

The eos_LoadTables routine fills a collection of data tables with the requested data tables from SESAME. Before calling this routine the host code must call eos_CreateTables to initialize memory for data tables and retrieve valid table handles. The host code may also need to call eos_SetOption, prior to calling eos_LoadTables, so the desired set up options can be changed from the documented defaults (see APPENDIX D).

The input arguments are:

nTables	This is the scalar EOS_INTEGER total number of data tables on which to operate.
tableHandles	This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i=1\dots nTables$. The host code is responsible for managing this array of table handles.

The output arguments are:

errorCode	This is a scalar EOS_INTEGER variable to contain an error code that may indicate failure to load the data. The host code must call eos_GetErrorCode and eos_GetErrorMessage to retrieve error details for a specified tableHandle. See APPENDIX G for error code details.
-----------	---

5.3.8 eos_SetDataFileName

NEW for 6.2.2 The eos_SetDataFileName routine is used to set the file name for a specified table handle. This will constrain EOSPAC to searching for applicable data within the specified file, if it's a valid file. See section 5.3.4 for details concerning the maximum length of the file name. This routine will fix an invalid table handle if it was invalidated by a previous call to eos_CreateTables. This routine must be called prior to eos_LoadTables for the specified table handle; otherwise an error will be returned. This routine should be used in conjunction with eos_GetMaxDataFileNameLength.

The input arguments are:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
matID	This is a scalar EOS_INTEGER containing the SESAME material identification number corresponding to the member data table.
tableType	This is a scalar EOS_INTEGER containing the table type corresponding to the member data table. See APPENDIX B and APPENDIX C for table type details.
fileName	This is a character string, of a maximum length defined by the constant named PATH_MAX, which is to contain the specified file name.

The output argument is:

errorCode This is a scalar EOS_INTEGER to contain an error code.

5.3.9 eos_SetPackedTables

The eos_SetPackedTables routine fills the specified data tables with data tables stored as a character array. Typically this is used to insert the data tables into the EOSPAC data structures after a multithread code has shared the data tables extracted by eos_GetPackedTables. The eos_SetPackedTables routine can also be used to unpack data tables recovered from a host code's binary restart file.

The input arguments are:

nTables This is the scalar EOS_INTEGER total number of data tables on which to operate.

packedTablesSize This is the scalar EOS_INTEGER number of bytes required to hold a specified list of data tables' member data – size of packedTables.

packedTables This is an array of EOS_CHAR that is used to store all of the member data of specified data tables. This array is designed to allow the host code to share data between multiple processors. If dynamic memory allocation for arrays is not possible, then eos_SetPackedTables will prove difficult to use since packedTables must hold packedTablesSize characters, where packedTablesSize is returned from the eos_GetPackedTablesSize routine.

The output arguments are:

tableHandles This is an array of EOS_INTEGER handles to particular data tables. Each handle corresponds to a member data table, T_i , where $i=1\dots nTables$. The host code is responsible for managing this array of table handles.

`errorCode` This is a scalar `EOS_INTEGER` to contain an error code that may indicate failure to unpack the data. The host code must call `eos_GetErrorCode` and `eos_GetErrorMessage` to retrieve error details for a specified `tableHandle`. See APPENDIX G for error code details.

5.4 C/C++ Language Bindings

```

void eos_CreateTables      ( EOS_INTEGER *nTables,
                           EOS_INTEGER tableType[],
                           EOS_INTEGER matID[],
                           EOS_INTEGER tableHandles[],
                           EOS_INTEGER *errorCode);

void eos_DestroyAll      ( EOS_INTEGER *errorCode);

void eos_DestroyTables   ( EOS_INTEGER *nTables,
                           EOS_INTEGER tableHandles[],
                           EOS_INTEGER *errorCode);

void eos_GetMaxDataFileNameLength ( EOS_INTEGER *max_length);

void eos_GetPackedTables ( EOS_INTEGER *nTables,
                           EOS_INTEGER tableHandles[],
                           EOS_CHAR *packedTables,
                           EOS_INTEGER *errorCode);

void eos_GetPackedTablesSize ( EOS_INTEGER *nTables,
                               EOS_INTEGER tableHandles[],
                               EOS_INTEGER *packedTablesSize,
                               EOS_INTEGER *errorCode);

void eos_GetTableCmnts   ( EOS_INTEGER *tableHandle,
                           EOS_CHAR *cmntStr,
                           EOS_INTEGER *errorCode);

void eos_GetTableInfo    ( EOS_INTEGER *tableHandle,
                           EOS_INTEGER *numInfoItems,
                           EOS_INTEGER infoItems[],
                           EOS_REAL infoVals[],
                           EOS_INTEGER *errorCode)

```

```

void eos_LoadTables      ( EOS_INTEGER *nTables,
                          EOS_INTEGER tableHandles[],
                          EOS_INTEGER *errorCode);

void eos_SetDataFileName ( EOS_INTEGER *tableHandle,
                          EOS_INTEGER *matID,
                          EOS_INTEGER *tableType,
                          EOS_CHAR *fileName,
                          EOS_INTEGER *errorCode);

void eos_SetPackedTables ( EOS_INTEGER *nTables,
                          EOS_INTEGER *packedTablesSize,
                          EOS_CHAR *packedTables,
                          EOS_INTEGER tableHandles[],
                          EOS_INTEGER *errorCode);

```

Use the header file named "eos_Interface.h" to define both the function prototypes listed above and the necessary constants used by EOSPAC. See section 9 for usage examples of these routines.

5.5 Fortran Language Bindings

```

subroutine eos_CreateTables      ( EOS_INTEGER nTables,
                                  EOS_INTEGER tableType(*),
                                  EOS_INTEGER matID(*),
                                  EOS_INTEGER tableHandles(*),
                                  EOS_INTEGER errorCode)

subroutine eos_DestroyAll      ( EOS_INTEGER errorCode)

subroutine eos_DestroyTables    ( EOS_INTEGER nTables,
                                  EOS_INTEGER tableHandles(*),
                                  EOS_INTEGER errorCode)

subroutine eos_GetMaxDataFileNameLength ( EOS_INTEGER max_length)

subroutine eos_GetPackedTables  ( EOS_INTEGER nTables,
                                  EOS_INTEGER tableHandles(*),
                                  EOS_CHAR packedTables,
                                  EOS_INTEGER errorCode)

```

```

subroutine eos_GetPackedTablesSize ( EOS_INTEGER nTables,
                                     EOS_INTEGER tableHandles(*),
                                     EOS_INTEGER packedTablesSize,
                                     EOS_INTEGER errorCode)

void eos_GetTableInfo                 ( EOS_INTEGER tableHandle,
                                     EOS_INTEGER numInfoltems,
                                     EOS_INTEGER infoltems,
                                     EOS_REAL infoVals,
                                     EOS_INTEGER errorCode);

subroutine eos_GetTableCmnts         ( EOS_INTEGER tableHandle,
                                     EOS_CHAR cmntStr,
                                     EOS_INTEGER errorCode)

subroutine eos_LoadTables            ( EOS_INTEGER nTables,
                                     EOS_INTEGER tableHandles(*),
                                     EOS_INTEGER errorCode)

subroutine eos_SetDataFileName       ( EOS_INTEGER tableHandle,
                                     EOS_INTEGER matID,
                                     EOS_INTEGER tableType,
                                     EOS_CHAR fileName,
                                     EOS_INTEGER errorCode)

subroutine eos_SetPackedTables       ( EOS_INTEGER nTables,
                                     EOS_INTEGER packedTablesSize,
                                     EOS_CHAR packedTables,
                                     EOS_INTEGER tableHandles(*),
                                     EOS_INTEGER errorCode)

```

Within a Fortran 77 host code, use the header file named “eos_Interface.fi” to define the necessary constants used by EOSPAC. See section 9 for Fortran 77 host code examples of using these routines.

Within a Fortran 90 host code, use the Fortran module named “eos_Interface” to define the necessary constants used by EOSPAC. See section 9 for Fortran 90 host code examples of using these routines.

6 INTERPOLATE MATERIAL DATA

The interpolation phase consists of calls to interface routines that use an established EOSPAC data table and return interpolated data requested by the host code. These routines are the only way to access the data tables.

6.1 Data Organization

Unlike the setup routines, the interpolation routines perform their function on data associated with a single table handle.

6.2 Routines and Parameters

The routines and parameters for interpolating material data are shown below.

6.2.1 eos_CheckExtrap

If the EOS_INTERP_EXTRAPOLATED error code is returned by either eos_Interpolate or eos_Mix, then the eos_CheckExtrap routine allows the user to determine which (x,y) pairs caused extrapolation and in which direction (high or low), it occurred. The units of the xVals, and yVals arguments listed below are determined by the units listed for each tableType in APPENDIX B and APPENDIX C.

The input arguments are

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
nXYPairs	This is the total number of pairs of independent variable values provided for interpolation for the specified table.
xVals	This is an array of the primary independent variable values to use during interpolation. There are nXYPairs elements in xVals.
yVals	This is an array of the secondary independent variable values to use during interpolation. There are nXYPairs elements in yVals.

The output arguments are

xyBounds	This is an array of size nXYPairs elements that returns EOS_OK if extrapolation did <u>not</u> occur. If extrapolation occurred the variable and direction are determined from Table 3.
errorCode	This is a scalar EOS_INTEGER to contain an error code.

In the case that eos_Mix returned EOS_INTERP_EXTRAPOLATED as an error code, an additional series of steps must be performed to determine which table handles correspond to the extrapolation error:

1. For each tableHandle sent to eos_Mix, call eos_GetErrorCode and, optionally, eos_GetErrorMessage.
2. For each of these tableHandles, call eos_CheckExtrap to determine one of codes listed in Table 3.

Table 3. Extrapolation return codes.

Code	Definition
EOS_OK	No extrapolation occurred.
EOS_xHi_yHi	Both the x and y arguments were high.
EOS_xHi_yOk	The x argument was high, the y argument was OK.
EOS_xHi_yLo	The x argument was high, the y argument was low. ⁴
EOS_xOk_yLo	The x argument is OK and the y argument is low. ⁴
EOS_xLo_yLo	Both the x and y arguments were low. ^{4,5}
EOS_xLo_yOk	The x argument was low, the y argument was OK. ⁵
EOS_xLo_yHi	The x argument was low, the y argument was OK. ⁵
EOS_xOk_yHi	The x argument is OK and the y argument is high.

⁴ If the y argument corresponds to a temperature value, then a zero temperature was used for interpolation rather than the value supplied by the host code.

⁵ If the x argument corresponds to a density value, then a zero density was used for interpolation rather than the value supplied by the host code.

Code	Definition
EOS_CANT_INVERT_DATA	Can't invert with respect to the required independent variable
EOS_CONVERGENCE_FAILED	Iterative algorithm did not converge during inverse interpolation
EOS_UNDEFINED	The result is undefined

6.2.2 eos_Interpolate

The eos_Interpolate routine provides interpolated values for a single material using a table handle associated with data stored within a data table. Before calling eos_Interpolate, the host code may need to call eos_SetOption so the desired interpolation options can be changed from the documented defaults. The units of the xVals, yVals, fVals, dFx and dFy arguments listed below are determined by the units listed for each tableType in APPENDIX B and APPENDIX C.

The input arguments are:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data. The host code is responsible for managing this table handle.
nXYPairs	This is the scalar EOS_INTEGER total number of pairs of independent variable values provided for interpolation.
xVals	This is an EOS_REAL array of the primary independent variable values to use during interpolation. There are nXYPairs elements in xVals.
yVals	This is an EOS_REAL array of the secondary independent variable values to use during interpolation. There are nXYPairs elements in yVals.

The output arguments are:

fVals	This is an EOS_REAL array of the interpolated data corresponding to the given independent variable data (x and y). There are nXYPairs elements in fVals.
-------	--

dFx	This is an EOS_REAL array of the interpolated partial derivatives of fVals with respect to x. There are nXYPairs elements in dFx.
dFy	This is an EOS_REAL array of the interpolated partial derivatives of fVals with respect to y. There are nXYPairs elements in dFy.
errorCode	This is a scalar EOS_INTEGER to contain an error code.

6.2.3 eos_Mix

The mixed material interpolation uses established EOSPAC data tables and returns interpolated data of mixed materials requested by the host code. The eos_Mix routine is the typical way to generate mixed material data using the data tables' member data tables. The data tables to be mixed must be of the same table type. An error code is returned if the table type is not valid for mixing (EOS_NullTable, EOS_Info, etc.). The table types that are valid for eos_Mix are limited to the following short list⁶ of 29 table types:

EOS_B_DT	EOS_Ktc_DT	EOS_Pt_DT	EOS_T_DUic	EOS_Uic_DT
EOS_Kc_DT	EOS_Pc_D	EOS_Pt_DUt	EOS_T_DUt	EOS_Ut_DPt
EOS_Kec_DT	EOS_Pe_DT	EOS_T_DPe	EOS_Uc_D	EOS_Ut_DT
EOS_Keo_DT	EOS_Pe_DUe	EOS_T_DPic	EOS_Ue_DPe	EOS_Zfc_DT
EOS_Kp_DT	EOS_Pic_DT	EOS_T_DPt	EOS_Ue_DT	EOS_Zfo_DT
EOS_Kr_DT	EOS_Pic_DUic	EOS_T_DUe	EOS_Uic_DPic	

The eos_Mix routine will provide interpolated values corresponding to mixtures of materials in pressure (or pressure and temperature) equilibrium. Before calling eos_Mix, the host code may need to call eos_SetOption so the desired interpolation and/or mixing options can be changed from the documented defaults. The units of the xVals, yVals, fVals, dFx and dFy arguments listed below are determined by the units listed for each tableType in APPENDIX B and APPENDIX C.

The input arguments are:

nTables	This is the total number of data tables on which to operate.
---------	--

⁶ These are cross-referenced to those of EOSPAC 5 within APPENDIX C.

tableHandles	This is an array of EOS_INTEGER handles to the tables to be mixed.
nXYPairs	This is the total number of pairs of independent variable values provided for interpolation for each table.
conclnMix	This is an EOS_REAL array containing the number fraction concentration corresponding to each independent variable value pair and to each tableHandle of the desired data to mix. There are nTables*nXYPairs elements in conclnMix, and it is stored sequentially in memory as follows: [conclnMix(i+(j-1)* nXYPairs): i=1 to nXYPairs], j=1 to nTables Note that the index, i, varies fastest as memory addresses increase incrementally.
xVals	This is an EOS_REAL array of the primary independent variable values to use during interpolation. There are nXYPairs elements in xVals.
yVals	This is an EOS_REAL array of the secondary independent variable values to use during interpolation. There are nXYPairs elements in yVals.

The output arguments are:

fVals	This is an EOS_REAL array of the interpolated data corresponding to the given independent variable data (x and y). There are nXYPairs elements in fVals.
dFx	This is an EOS_REAL array of the interpolated partial derivatives of fVals with respect to x. There are nXYPairs elements in dFx.
dFy	This is an EOS_REAL array of the interpolated partial derivatives of fVals with respect to y. There are nXYPairs elements in dFy.
errorCode	This is a scalar EOS_INTEGER to contain an error code.

6.3 C/C++ Language Bindings

```

void eos_CheckExtrap      ( EOS_INTEGER *tableHandle,
                           EOS_INTEGER *nXYPairs,
                           EOS_REAL *xVals,
                           EOS_REAL *yVals,
                           EOS_INTEGER *xyBounds,
                           EOS_INTEGER *errorCode);

void eos_Interpolate      ( EOS_INTEGER *tableHandle,
                           EOS_INTEGER *nXYPairs,
                           EOS_REAL *xVals,
                           EOS_REAL *yVals,
                           EOS_REAL *fVals,
                           EOS_REAL *dFx,
                           EOS_REAL *dFy,
                           EOS_INTEGER *errorCode);

void eos_Mix               ( EOS_INTEGER *nTables,
                           EOS_INTEGER *tableHandles,
                           EOS_INTEGER *nXYPairs,
                           EOS_REAL *conclnMix,
                           EOS_REAL *xVals,
                           EOS_REAL *yVals,
                           EOS_REAL *fVals,
                           EOS_REAL *dFx,
                           EOS_REAL *dFy,
                           EOS_INTEGER *errorCode);

```

Use the header file named "eos_Interface.h" to define both the function prototypes listed above and the necessary constants used by EOSPAC. See section 9 for usage examples of these routines.

6.4 Fortran Language Bindings

```
subroutine eos_CheckExtrap      ( EOS_INTEGER tableHandle,  
                                EOS_INTEGER nXYPairs,  
                                EOS_REAL xVals(*),  
                                EOS_REAL yVals(*),  
                                EOS_INTEGER xyBounds(*),  
                                EOS_INTEGER errorCode)  
  
subroutine eos_Interpolate    ( EOS_INTEGER tableHandle,  
                                EOS_INTEGER nXYPairs,  
                                EOS_REAL xVals(*),  
                                EOS_REAL yVals(*),  
                                EOS_REAL fVals(*),  
                                EOS_REAL dFx(*),  
                                EOS_REAL dFy(*),  
                                EOS_INTEGER errorCode)  
  
subroutine eos_Mix            ( EOS_INTEGER nTables,  
                                EOS_INTEGER tableHandles(*),  
                                EOS_INTEGER nXYPairs,  
                                EOS_REAL conclnMix(*),  
                                EOS_REAL xVals(*),  
                                EOS_REAL yVals(*),  
                                EOS_REAL fVals(*),  
                                EOS_REAL dFx(*),  
                                EOS_REAL dFy(*),  
                                EOS_INTEGER errorCode)
```

Within a Fortran 77 host code, use the header file named "eos_Interface.fi" to define the necessary constants used by EOSPAC. See section 9 for Fortran 77 host code usage examples of these routines.

Within a Fortran 90 host code, use the Fortran module named "eos_Interface" to define the necessary constants used by EOSPAC. See section 9 for Fortran 90 host code usage examples of these routines.

7 MISCELLANEOUS INFORMATION ROUTINES

This section provides descriptions of some routines that submit or return miscellaneous information about or related to a data table or its contents. These routines are the only way to set or retrieve this information.

7.1 Routines and Parameters

The routines and parameters that provide miscellaneous information are shown below.

7.1.1 eos_GetErrorCode

The eos_GetErrorCode routine is used to the most recent EOSPAC error code that corresponds to a specific table handle.

The input argument is:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
-------------	--

The output argument is

errorCode	This is a scalar EOS_INTEGER to contain the requested error code.
-----------	---

7.1.2 eos_GetErrorMessage

The eos_GetErrorMessage routine is used to translate an EOSPAC error code (see APPENDIX G) into an informative text message.

The input argument is:

errorCode	This is a scalar EOS_INTEGER to contain an error code.
-----------	--

The output argument is

errorMsg	This is a character string of a maximum length defined by the constant named EOS_MaxErrMsgLen.
----------	--

7.1.3 eos_GetTableCmnts

The eos_GetTableCmnts routine returns the comments available about the requested data table. Unlike most other routines in this section, eos_GetTableCmnts works on only a single data table.

Before calling eos_GetTableCmnts, the host code must call eos_GetTableInfo to find out the length of the comments, lenCmnts, allowing the host code to allocate adequate storage.

The input arguments are:

tableHandle This is the scalar EOS_INTEGER handle to particular data table.

The output arguments are:

cmntStr This is a string of EOS_CHAR, of length lenCmnts, containing the requested comments. The value of lenCmnts for each tableHandle can be obtained by calling eos_GetTableInfo using the constant named EOS_Cmnt_Len (see APPENDIX E for details). If dynamic memory allocation for strings is not possible, then eos_GetTableCmnts will prove difficult to use.

errorCode This is a scalar EOS_INTEGER variable to contain an error code that may indicate the comment table(s) could not be loaded. The host code must call eos_GetErrorCode and eos_GetErrorMessage to retrieve error details for a specified tableHandle. See APPENDIX G for error code details.

7.1.4 eos_GetTableInfo

The eos_GetTableInfo routine returns the values of requested information about data table members. Unlike most other "Data Setup" routines, this routine works on only a single data table. Information is requested by passing a list of parameters to the routine that returns the requested information in the same order. The information that can be requested is in APPENDIX E.

The input arguments are:

tableHandle	This is the EOS_INTEGER handle to particular data table.
numInfoItems	EOS_INTEGER scalar number of information items requested.
infoItems	This is an EOS_INTEGER array of information items requested. The allowed values are in APPENDIX E.

The output arguments are:

infoVals	This is an EOS_REAL array containing the information items requested. It contains numInfoItems values. The values are in the same order as requested in the infoItems array.
errorCode	This is a scalar EOS_INTEGER to contain the error code. See APPENDIX G for error code details.

7.1.5 eos_GetVersion

The eos_GetVersion routine is used to retrieve a character string defining the current version of EOSPAC.

There are no input arguments.

The output argument is:

Version	This is a character string of a maximum length defined by the value returned by eos_GetVersionLength.
---------	---

7.1.6 eos_GetVersionLength

The eos_GetVersionLength routine is used to retrieve the length of the string returned by eos_GetVersion.

There are no input arguments.

The output argument is:

Length	This is a scalar EOS_INTEGER defining the length of the string returned by eos_GetVersion. This length includes the null ('\0') terminating character, which is used in the "C" programming language.
--------	---

7.1.7 eos_ResetOption

The eos_ResetOption routine is used to reset an option related to a specified table handle to its default state (see APPENDIX D and APPENDIX F for default settings). The eos_ResetOption routine is used prior to calling eos_LoadTables, eos_Interpolate, and/or eos_Mix to specify applicable options for each table handle.

The input arguments are:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
tableOption	This is a scalar EOS_INTEGER containing the option flag indicating what option to set corresponding to the tableHandle. See APPENDIX D and APPENDIX F for table option details.

The output argument is:

errorCode	This is a scalar EOS_INTEGER to contain an error code.
-----------	--

7.1.8 eos_SetOption

The eos_SetOption routine is used to set an option related to a specified table handle. The eos_SetOption routine is used prior to calling eos_LoadTables, eos_Interpolate, and/or eos_Mix to specify applicable options for each table handle.

The input arguments are:

tableHandle	This is a scalar EOS_INTEGER handle to a particular data table. The host code is responsible for managing this table handle.
tableOption	This is a scalar EOS_INTEGER containing the option flag indicating what option to set corresponding to the tableHandle. See APPENDIX D and APPENDIX F for table option details.
tableOptionVal	This is a scalar EOS_REAL containing the option value to be assigned to the tableHandle. Note that not all of the option flags defined in APPENDIX D and APPENDIX F use this value; however, a variable or literal is required when calling eos_SetOption due to the limitations of a flat public interface.

The output argument is:

errorCode	This is a scalar EOS_INTEGER to contain an error code.
-----------	--

7.2 C/C++ Language Bindings

```

void eos_GetMaxDataFileNameLength ( EOS_INTEGER *max_length);
void eos_GetErrorCode ( EOS_INTEGER *tableHandle,
                       EOS_INTEGER *errorCode);
void eos_GetErrorMessage ( EOS_INTEGER *errorCode,
                           EOS_CHAR errorMsg[EOS_MaxErrMsgLen]);
void eos_GetVersion ( EOS_CHAR *version);
void eos_GetVersionLength ( EOS_INTEGER *length);
void eos_ResetOption ( EOS_INTEGER *tableHandle,
                      const EOS_INTEGER *tableOption,
                      EOS_INTEGER *errorCode);
void eos_SetDataFileName ( EOS_INTEGER *tableHandle,
                           EOS_CHAR *fileName,
                           EOS_INTEGER *errorCode);

```

```
void eos_SetOption      ( EOS_INTEGER *tableHandle,
                        const EOS_INTEGER *tableOption,
                        const EOS_REAL *tableOptionVal,
                        EOS_INTEGER *errorCode);
```

Use the header file named "eos_Interface.h" to define both the function prototypes listed above and the necessary constants used by EOSPAC. See section 9 for usage examples of these routines.

7.3 Fortran Language Bindings

```
subroutine eos_GetMaxDataFileNameLength  ( EOS_INTEGER max_length)

subroutine eos_GetErrorCode              ( EOS_INTEGER tableHandle,
                                          EOS_INTEGER errorCode)

subroutine eos_GetErrorMessage           ( EOS_INTEGER errorCode,
                                          EOS_CHAR
                                          errorMsg(EOS_MaxErrMsgLen))

subroutine eos_GetVersion                ( EOS_CHAR version)

subroutine eos_GetVersionLength          ( EOS_INTEGER length)

subroutine eos_ResetOption               ( EOS_INTEGER tableHandle,
                                          EOS_INTEGER tableOption,
                                          EOS_INTEGER errorCode)

subroutine eos_SetDataFileName           ( EOS_INTEGER tableHandle,
                                          EOS_CHAR fileName,
                                          EOS_INTEGER errorCode)

subroutine eos_SetOption                  ( EOS_INTEGER tableHandle,
                                          const EOS_INTEGER tableOption,
                                          const EOS_REAL tableOptionVal,
                                          EOS_INTEGER errorCode)
```

Within a Fortran 77 host code, use the header file named "eos_Interface.fi" to define the necessary constants used by EOSPAC. See section 9 for Fortran 77 host code usage examples of these routines.

Within a Fortran 90 host code, use the Fortran module named "eos_Interface" to define the necessary constants used by EOSPAC. See section 9 for Fortran 90 host code usage examples of these routines.

8 SELECTED NUMERIC DETAILS

This section provides additional descriptions of some complex EOSPAC features, which are implemented to address some numeric issue or other.

8.1 Custom Smoothing and Interpolation

At the request of the user community, some very specific data smoothing capabilities of SAGE⁷ have been added to EOSPAC. These features correspond to the setup and interpolation options EOS_PT_SMOOTHING, EOS_ADJUST_VAP_PRES, and EOS_USE_CUSTOM_INTERP. When the setup option, EOS_PT_SMOOTHING, is enabled for a table handle, the loaded equation of state data is smoothed in preparation for using the EOS_USE_CUSTOM_INTERP interpolation option. The setup option, EOS_ADJUST_VAP_PRES, is provided as a mechanism to shift the vapor pressure data according to

$$P_i = P_i - P_{\text{shift}} \cdot \left(1 - \frac{P_i}{P_{i-1}} \right), \quad (8.1)$$

where P_{shift} is the user-provided pressure value (GPa) that is used to ensure the ambient conditions of the tabulated data are acceptable. This vapor pressure adjustment has often been deemed necessary, and it is material-dependent.

Once the data has been loaded and smoothed according to the rules associated with EOS_PT_SMOOTHING and EOS_ADJUST_VAP_PRES, interpolation may be performed with the EOS_USE_CUSTOM_INTERP option set. This interpolation option is limited to use with the EOS_Ut_PtT and EOS_V_PtT data types (pressure- and temperature-dependent internal-energy and specific volume respectively). The EOS_USE_CUSTOM_INTERP interpolation option uses linear interpolation to calculate the desired values from isotherms, which contain data made to conform to

⁷ SAGE is a one-, two-, and three-dimensional, multi-material Eulerian hydrodynamics code (LA-UR-04-2959).

Maxwell's relations (Maxwell Construction, or Equal Area Construction). Any basic thermodynamics textbook should contain a description of Maxwell's relations.

8.2 Forced Data Monotonicity

Much of the data in the SESAME database is not monotonic with respect to one or both of the tabulated independent variables. This is a problem when a data table is to be inverted with respect to one of the tabulated independent variables. To ensure either global increasing- or decreasing-monotonicity, a simple algorithm is used, in which the average of a function's local minimum and local maximum is determined and then used to replace the local tabulated function values. Once that is done monotonicity is achieved, but a small slope is then imposed over the localized region so that either global increasing- or decreasing-monotonicity is imposed. The aforementioned small slope is calculated to be three-orders-of-magnitude larger than the machine's floating point precision (i.e., $\sim 10^{-12}$ on a 64-bit IEEE machine). It is important to note that this forced data monotonicity algorithm is not an "equal-area" calculation, which is used to impose Maxwell constructions on an EOS.

Figure 9 graphically describes the result (orange line) of this algorithm when applied to an isotherm (blue line) of an arbitrary pressure function. Although it cannot be seen due to the plot's pressure range, the orange line actually has an artificial slope of approximately 10^{-12} in the nearly-horizontal region, where the left-most pressure value of said region is the average of P_{min} and P_{max} .

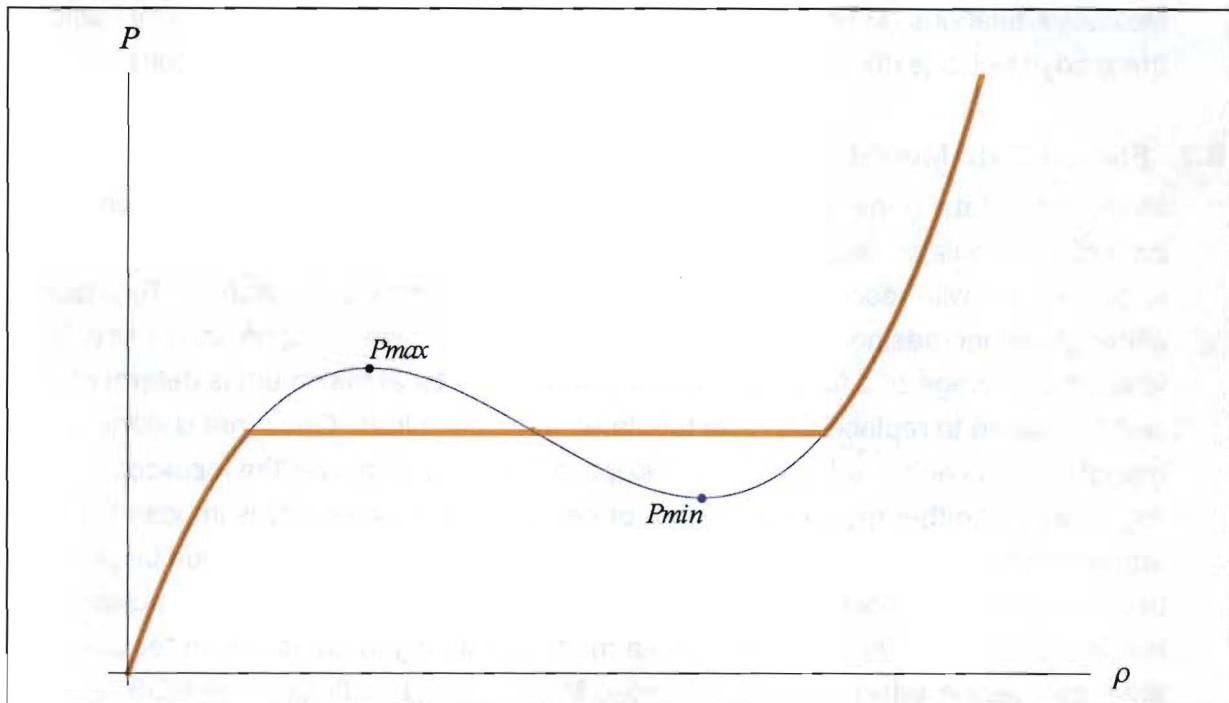


Figure 9. General depiction of $P(\rho)$ forced to be monotonically-increasing.

8.3 Extended Precision is Disabled

In an effort to improve the portability of EOSPAC, the extended precision features of some machine architectures are disabled upon entry to any of EOSPAC's public routines, and then the extended precision is re-enabled prior to exiting said public routines. The problem of extended precision is described as follows [11]:

The IEEE-754 standard defines the bit-level behavior of floating-point arithmetic operations on all modern processors. This allows numerical programs to be ported between different platforms with identical results, in principle. In practice, there are often minor variations caused by differences in the order of operations (depending on the compiler and optimization level) but these are generally not significant.

However, more noticeable discrepancies can be seen when porting numerical programs between x86 systems and other platforms, because the the x87 floating point unit (FPU) on x86 processors computes results using extended precision internally (the values being converted to double precision only when they are stored to memory). In contrast, processors such as SPARC, PA-RISC,

Alpha, MIPS and POWER/PowerPC work with native double-precision values throughout. The differences between these implementations lead to changes in rounding and underflow/overflow behavior, because intermediate values have a greater relative precision and exponent range when computed in extended precision. In particular, comparisons involving extended precision values may fail where the equivalent double precision values would compare equal.

To avoid these incompatibilities, the x87 FPU also offers a hardware double-precision rounding mode. In this mode the results of each extended-precision floating-point operation are rounded to double precision in the floating-point registers by the FPU. It is important to note that the rounding only affects the precision, not the exponent range, so the result is a hybrid double-precision format with an extended range of exponents. On BSD systems such as FreeBSD, NetBSD and OpenBSD, the hardware double-precision rounding mode is the default, giving the greatest compatibility with native double precision platforms. On x86 GNU/Linux systems the default mode is extended precision (with the aim of providing increased accuracy). To enable the double-precision rounding mode it is necessary to override the default setting on per-process basis using the FLDCW "floating-point load control-word" machine instruction.

As a result of the problem described above, every effort is made to disable extended precision arithmetic on x86 machines.

8.4 Mass Fraction Data Interpolation

For selected materials, Sesame contains mass fraction data tables, which tabulate phase-specific (i.e., beta, gamma, liquid, etc.) mass fraction data. EOSPAC has the capability to access and interpolate this mass fraction data if it's available. To implement this capability while minimizing changes to the public interface specification, the `eos_GetTableInfo` function (section 7.1.4) is used with `eos_Interpolate` (section 6.2.2) in an unusual way. Once the material data is loaded into memory using the `EOS_M_DT` data type option, the host code must call `eos_GetTableInfo` to obtain the total number of tabulated phases (see the `EOS_NUM_PHASES` parameter in APPENDIX E). Then the `eos_Interpolate` output array (`fVals`) must be allocated so that all of the material's phases can be interpolated at once; however, allocation of the derivative arrays (`dFx` and `dFy`) is not required since they are ignored within EOSPAC. For example, if `nXYPairs` is set to ten and the number of phases is three, then the output array for `eos_Interpolate` are each allocated to hold thirty values; whereas, the input arrays (`xVals` and `yVals`)

are only allocated to hold ten values. The interpolated output is organized so that each phase's interpolated mass fractions are stored in turn. The following "C" code snippet demonstrates how the input and output arrays are organized:

```
xVals = (EOS_REAL *) malloc (sizeof (EOS_REAL) * nXYPairs);
yVals = (EOS_REAL *) malloc (sizeof (EOS_REAL) * nXYPairs);
fVals = (EOS_REAL *) malloc (sizeof (EOS_REAL) * num_phases * nXYPairs);
for (j = 0; j < num_phases; j++) {
  for (k = 0; k < nXYPairs; k++) {
    printf ("%23.15e %23.15e %23.15e\n",
            xVals[k], yVals[k], fVals[nXYPairs*j + k]);
  }
}
```

To maintain data integrity, the interpolation is limited to use only the bilinear (EOS_LINEAR) interpolator for the EOS_M_DT data type.

8.5 Numerical Integration

The capability to calculate entropy data is implemented with multiple algorithms. One such algorithm depends upon the numerical integration of the tabulated internal energy data with respect to temperature. To perform the numerical integration, a simple trapezoid rule is implemented. A specific note of interest is that the trapezoid integration equally-divides each tabulated temperature interval into ninety-nine sub-intervals prior to interpolation. The hard-wired number of sub-intervals was chosen arbitrarily because it seemed adequate. Another, but more subtle, item to note is that the form of equation (2.2) is implemented within EOSPAC so that the integrand values, $\frac{u}{T^2}$, for all applicable tabulated data are passed to the interpolator within the trapezoid integration algorithm instead of interpolating the internal energy, $u = u(\rho, T)$, prior to calculating the integrand. This smoothes the calculated results by damping incurred numerical errors.

9 USAGE EXAMPLES

This section contains various examples showing the usage of the interface routines defined in Sections 5 through 7.

9.1 C Host Code Example

EOSPAC User's Manual: Version 6.2

```

/*****
 * Example Program
 * -----
 * Filetype: (SOURCE)
 *
 * Copyright -- see file named COPYRIGHTNOTICE
 *
 *****/

/#! \file
 * \ingroup examples
 * \brief This is a simple C example of how to use EOSPAC6 interface.
 */

#include <stdio.h>
#include <stdlib.h>
#include "eos_Interface.h"

int main ()
{
    enum
    { nTablesE = 5 };
    enum
    { nXYPairsE = 4 };
    enum
    { nInfoItemsE = 12 };

    EOS_INTEGER i, j;
    EOS_REAL X[nXYPairsE], Y[nXYPairsE], F[nXYPairsE], dFx[nXYPairsE],
        dFy[nXYPairsE];
    EOS_INTEGER tableType[nTablesE], numIndVars[nTablesE];
    EOS_INTEGER matID[nTablesE];
    EOS_INTEGER tableHandle[nTablesE];
    EOS_INTEGER errorCode;
    EOS_INTEGER tableHandleErrorCode;
    EOS_INTEGER nTables;
    EOS_INTEGER nXYPairs;
    EOS_REAL infoVals[nInfoItemsE];
    EOS_INTEGER nInfoItems;
    EOS_INTEGER infoItems[nInfoItemsE] = {
        EOS_Cmnt_Len,
        EOS_Exchange_Coeff,
        EOS_F_Convert_Factor,
        EOS_Log_Val,
        EOS_Material_ID,
        EOS_Mean_Atomic_Mass,
        EOS_Mean_Atomic_Num,
        EOS_Modulus,
        EOS_Normal_Density,
        EOS_Table_Type,
        EOS_X_Convert_Factor,
        EOS_Y_Convert_Factor
    };
    EOS_CHAR *infoItemDescriptions[nInfoItemsE] = {

```

```

    "The length in characters of the comments available for the specified
    data table",
    "The exchange coefficient",
    "The conversion factor corresponding to the dependent variable,
    F(x,y)",
    "Non-zero if the data table is in a log10 format",
    "The SESAME material identification number",
    "The mean atomic mass",
    "The mean atomic number",
    "The solid bulk modulus",
    "The normal density",
    "The type of data table. Corresponds to the parameters in APPENDIX B
    and APPENDIX C",
    "The conversion factor corresponding to the primary independent
    variable, x",
    "The conversion factor corresponding to the secondary independent
    variable, y"
};
EOS_CHAR *tableTypeLabel[nTablesE] = {
    "EOS_Pt_DT",
    "EOS_Dv_T",
    "EOS_Ogb",
    "EOS_Comment",
    "EOS_Info"
};
EOS_CHAR errorMessage[EOS_MaxErrMsgLen];

EOS_INTEGER one = 1;

nTables = nTablesE;
nXYPairs = nXYPairsE;
nInfoItems = nInfoItemsE;

/*
 * EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
 * EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
 * EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
 * EOS_Comment, material 2140 works for Sesame tables 101-199 (record
type 4)
 * EOS_Info, material 2140 works for Sesame table 201 (record type 5)
 */
tableType[0] = EOS_Pt_DT;
tableType[1] = EOS_Dv_T;
tableType[2] = EOS_Ogb;
tableType[3] = EOS_Comment;
tableType[4] = EOS_Info;

numIndVars[0] = 2;
numIndVars[1] = 1;
numIndVars[2] = 0;
numIndVars[3] = 0;
numIndVars[4] = 0;

matID[0] = 2140;

```

EOSPAC User's Manual: Version 6.2

```
matID[1] = 2140;
matID[2] = 12140;
matID[3] = 2140;
matID[4] = 2140;

errorCode = EOS_OK;
for (i = 0; i < nTables; i++) {
    tableHandle[i] = 0;
}

/*
 * initialize table data objects
 */

eos_CreateTables (&nTables, tableType, matID, tableHandle, &errorCode);
if (errorCode != EOS_OK) {
    for (i = 0; i < nTables; i++) {
        tableHandleErrorCode = EOS_OK;
        eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
        eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
        printf ("eos_CreateTables ERROR %i: %s\n", tableHandleErrorCode,
                errorMessage);
    }
}

/*
 * set some options
 */

for (i = 0; i < nTables; i++) {
    /* enable smoothing */
    eos_SetOption (&tableHandle[i], &EOS_SMOOTH, EOS_NullPtr, &errorCode);
    if (errorCode != EOS_OK) {
        eos_GetErrorMessage (&errorCode, errorMessage);
        printf ("eos_SetOption ERROR %i: %s\n", errorCode, errorMessage);
    }
}

/*
 * load data into table data objects
 */

eos_LoadTables (&nTables, tableHandle, &errorCode);
if (errorCode != EOS_OK) {
    eos_GetErrorMessage (&errorCode, errorMessage);
    printf ("eos_LoadTables ERROR %i: %s\n", errorCode, errorMessage);
    for (i = 0; i < nTables; i++) {
        tableHandleErrorCode = EOS_OK;
        eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
        eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
        printf ("eos_LoadTables ERROR %i (TH=%i): %s\n",
                tableHandleErrorCode,
                tableHandle[i], errorMessage);
    }
}
```

```

}

/*
 * interpolate -- errors codes are intentionally produced
 */
X[0] = 3000.;
X[1] = 6000.;
X[2] = 8200.;
X[3] = 8300.;

Y[0] = 20000.0;
Y[1] = 620000.0;
Y[2] = 4000000.0;
Y[3] = 200000000.0;

for (i = 0; i < nTables; i++) {
    printf ("\n--- Interpolate using tableType %s ---\n",
tableTypeLabel[i]);
    eos_Interpolate (&tableHandle[i], &nXYPairs, X, Y, F, dFx, dFy,
&errorCode);
    printf ("%s Interpolation Results:\n", tableTypeLabel[i]);
    if (errorCode != EOS_OK) {
        eos_GetErrorMessage (&errorCode, errorMessage);
        printf ("eos_Interpolate ERROR %i (TH=%i): %s\n", errorCode,
tableHandle[i], errorMessage);
    }
    else {
        for (j = 0; j < nXYPairs; j++) {
            if (numIndVars[i] == 1)
                printf ("\ti=%i\tX = %e, F = %e, dFx = %e, errorCode: %d\n",
j, X[j], F[j], dFx[j], errorCode);
            if (numIndVars[i] == 2)
                printf
                ("\ti=%i\tX = %e, Y = %e, F = %e, dFx = %e, dFy = %e,
errorCode: %d\n",
j, X[j], Y[j], F[j], dFx[j], dFy[j], errorCode);
        }
    }
}

/*
 * retrieve table info -- errors codes are intentionally produced
 */

for (i = 0; i < nTables; i++) {
    printf ("\n--- Table information for tableType %s , tableHandle=%i ---
\n",
tableTypeLabel[i], tableHandle[i]);
    for (j = 0; j < nInfoItems; j++) {
        eos_GetTableInfo (&(tableHandle[i]), &one, &(infoItems[j]),
&(infoVals[j]), &errorCode);
        if (errorCode == EOS_OK) {
            printf ("%2i. %-82s: %13.6f\n", j + 1, infoItemDescriptions[j],
infoVals[j]);
        }
    }
}

```

```

    }
    else if (errorCode != EOS_INVALID_INFO_FLAG) {
        /* Ignore EOS_INVALID_INFO_FLAG since not all infoItems are
        currently
        applicable to a specific tableHandle. */
        eos_GetErrorMessage (&errorCode, errorMessage);
        printf ("eos_GetTableInfo ERROR %i: %s\n", errorCode,
        errorMessage);
    }
}
}

/*
 * Destroy all data objects
 */

eos_DestroyAll (&errorCode);
if (errorCode != EOS_OK) {
    for (i = 0; i < nTables; i++) {
        tableHandleErrorCode = EOS_OK;
        eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
        eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
        printf ("eos_DestroyAll ERROR %i: %s\n", tableHandleErrorCode,
        errorMessage);
    }
}

return 0;
}

```

9.2 C++ Host Code Example

```

| /*****
| * Example Program
| * -----
| * Filetype: (SOURCE)
| *
| * Copyright -- see file named COPYRIGHTNOTICE
| *
| *****/
|
|/! \file
| * \ingroup examples
| * \brief This is a simple C++ example of how to use EOSPAC6 interface.
| */
|
#include <iostream>
#include "eos_Interface.h"

```

```

using namespace std;

int main ()
{
    const EOS_INTEGER nTablesE = 5;
    const EOS_INTEGER nXYPairsE = 4;
    const EOS_INTEGER nInfoItemsE = 12;

    EOS_INTEGER i, j;
    EOS_REAL X[nXYPairsE], Y[nXYPairsE], F[nXYPairsE], dFx[nXYPairsE],
        dFy[nXYPairsE];
    EOS_INTEGER tableType[nTablesE], numIndVars[nTablesE];
    EOS_INTEGER matID[nTablesE];
    EOS_INTEGER tableHandle[nTablesE];
    EOS_INTEGER errorCode;
    EOS_INTEGER tableHandleErrorCode;
    EOS_INTEGER nTables;
    EOS_INTEGER nXYPairs;
    EOS_REAL infoVals[nInfoItemsE];
    EOS_INTEGER nInfoItems;
    EOS_INTEGER infoItems[nInfoItemsE] = {
        EOS_Cmnt_Len,
        EOS_Exchange_Coeff,
        EOS_F_Convert_Factor,
        EOS_Log_Val,
        EOS_Material_ID,
        EOS_Mean_Atomic_Mass,
        EOS_Mean_Atomic_Num,
        EOS_Modulus,
        EOS_Normal_Density,
        EOS_Table_Type,
        EOS_X_Convert_Factor,
        EOS_Y_Convert_Factor
    };
    const EOS_CHAR *infoItemDescriptions[nInfoItemsE] = {
        "The length in characters of the comments available for the specified
data table",
        "The exchange coefficient",
        "The conversion factor corresponding to the dependent variable,
F(x,y)",
        "Non-zero if the data table is in a log10 format",
        "The SESAME material identification number",
        "The mean atomic mass",
        "The mean atomic number",
        "The solid bulk modulus",
        "The normal density",
        "The type of data table. Corresponds to the parameters in APPENDIX B
and APPENDIX C",
        "The conversion factor corresponding to the primary independent
variable, x",
        "The conversion factor corresponding to the secondary independent
variable, y"
    };
};

```

EOSPAC User's Manual: Version 6.2

```
const EOS_CHAR *tableTypeLabel[nTablesE] = {
    "EOS_Pt_DT",
    "EOS_Dv_T",
    "EOS_Ogb",
    "EOS_Comment",
    "EOS_Info"
};
EOS_CHAR errorMessage[EOS_MaxErrMsgLen];

EOS_INTEGER one = 1;

nTables = nTablesE;
nXYPairs = nXYPairsE;
nInfoItems = nInfoItemsE;

/*
 * EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
 * EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
 * EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
 * EOS_Comment, material 2140 works for Sesame tables 101-199 (record
type 4)
 * EOS_Info, material 2140 works for Sesame table 201 (record type 5)
 */
tableType[0] = EOS_Pt_DT;
tableType[1] = EOS_Dv_T;
tableType[2] = EOS_Ogb;
tableType[3] = EOS_Comment;
tableType[4] = EOS_Info;

numIndVars[0] = 2;
numIndVars[1] = 1;
numIndVars[2] = 0;
numIndVars[3] = 0;
numIndVars[4] = 0;

matID[0] = 2140;
matID[1] = 2140;
matID[2] = 12140;
matID[3] = 2140;
matID[4] = 2140;

errorCode = EOS_OK;
for (i = 0; i < nTables; i++) {
    tableHandle[i] = 0;
}

/*
 * initialize table data objects
 */

eos_CreateTables (&nTables, tableType, matID, tableHandle, &errorCode);
if (errorCode != EOS_OK) {
    for (i = 0; i < nTables; i++) {
        tableHandleErrorCode = EOS_OK;
    }
}
```

```

        eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
        eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
        printf ("eos_CreateTables ERROR %i: %s\n", tableHandleErrorCode,
                errorMessage);
    }
}

/*
 * set some options
 */

for (i = 0; i < nTables; i++) {
    /* enable smoothing */
    eos_SetOption (&tableHandle[i], &EOS_SMOOTH, EOS_NullPtr, &errorCode);
    if (errorCode != EOS_OK) {
        eos_GetErrorMessage (&errorCode, errorMessage);
        printf ("eos_SetOption ERROR %i: %s\n", errorCode, errorMessage);
    }
}

/*
 * load data into table data objects
 */

eos_LoadTables (&nTables, tableHandle, &errorCode);
if (errorCode != EOS_OK) {
    eos_GetErrorMessage (&errorCode, errorMessage);
    printf ("eos_LoadTables ERROR %i: %s\n", errorCode, errorMessage);
    for (i = 0; i < nTables; i++) {
        tableHandleErrorCode = EOS_OK;
        eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
        eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
        printf ("eos_LoadTables ERROR %i (TH=%i): %s\n",
                tableHandleErrorCode,
                tableHandle[i], errorMessage);
    }
}

/*
 * interpolate -- errors codes are intentionally produced
 */
X[0] = 3000.;
X[1] = 6000.;
X[2] = 8200.;
X[3] = 8300.;

Y[0] = 20000.0;
Y[1] = 620000.0;
Y[2] = 4000000.0;
Y[3] = 200000000.0;

for (i = 0; i < nTables; i++) {
    printf ("\n--- Interpolate using tableType %s ---\n",
            tableTypeLabel[i]);
}

```

EOSPAC User's Manual: Version 6.2

```
    eos_Interpolate (&tableHandle[i], &nXYPairs, X, Y, F, dFx, dFy,
                    &errorCode);
    printf ("%s Interpolation Results:\n", tableTypeLabel[i]);
    if (errorCode != EOS_OK) {
        eos_GetErrorMessage (&errorCode, errorMessage);
        printf ("eos_Interpolate ERROR %i (TH=%i): %s\n", errorCode,
                tableHandle[i], errorMessage);
    }
    else {
        for (j = 0; j < nXYPairs; j++) {
            if (numIndVars[i] == 1)
                printf ("\ti=%i\tX = %e, F = %e, dFx = %e, errorCode: %d\n",
                        j, X[j], F[j], dFx[j], errorCode);
            if (numIndVars[i] == 2)
                printf
                    ("\ti=%i\tX = %e, Y = %e, F = %e, dFx = %e, dFy = %e,
errorCode: %d\n",
                    j, X[j], Y[j], F[j], dFx[j], dFy[j], errorCode);
        }
    }
}
}

/*
 * retrieve table info -- errors codes are intentionally produced
 */

for (i = 0; i < nTables; i++) {
    printf ("\n--- Table information for tableType %s , tableHandle=%i ---
\n",
            tableTypeLabel[i], tableHandle[i]);
    for (j = 0; j < nInfoItems; j++) {
        eos_GetTableInfo (&(tableHandle[i]), &one, &(infoItems[j]),
                        &(infoVals[j]), &errorCode);
        if (errorCode == EOS_OK) {
            printf ("%2i. %-82s: %13.6f\n", j + 1, infoItemDescriptions[j],
                    infoVals[j]);
        }
        else if (errorCode != EOS_INVALID_INFO_FLAG) {
            /* Ignore EOS_INVALID_INFO_FLAG since not all infoItems are
currently
            applicable to a specific tableHandle. */
            eos_GetErrorMessage (&errorCode, errorMessage);
            printf ("eos_GetTableInfo ERROR %i: %s\n", errorCode,
                    errorMessage);
        }
    }
}

/*
 * Destroy all data objects
 */

eos_DestroyAll (&errorCode);
if (errorCode != EOS_OK) {
```

```

    for (i = 0; i < nTables; i++) {
        tableHandleErrorCode = EOS_OK;
        eos_GetErrorCode (&tableHandle[i], &tableHandleErrorCode);
        eos_GetErrorMessage (&tableHandleErrorCode, errorMessage);
        printf ("eos_DestroyAll ERROR %i: %s\n", tableHandleErrorCode,
                errorMessage);
    }
}

return 0;
}

```

9.3 Fortran 77 Host Code Example

```

c*****
c Example F77 Program
c -----
c Filetype: (SOURCE)
c
c Copyright -- see file named COPYRIGHTNOTICE
c
c*****

c> \file
c> \ingroup examples
c> \brief This is a simple F77 example of how to use EOSPAC6 interface.

```

```

    program TestF77

    implicit none

    include 'eos_Interface.fi'

    integer*4 nTables, nXYPairs, nInfoItems
    parameter (nTables = 5)
    parameter (nXYPairs = 4)
    parameter (nInfoItems = 12)

    integer*4 i, j
    real*8 X(nXYPairs), Y(nXYPairs), F(nXYPairs), dFx(nXYPairs),
    &      dFy(nXYPairs)
    integer*4 tableType(nTables), numIndVars(nTables)
    integer*4 matID(nTables)
    integer*4 tableHandle(nTables)
    integer*4 errorCode
    integer*4 tableHandleErrorCode
    real*8 infoVals(nInfoItems)
    integer*4 infoItems(nInfoItems)
    character*82 infoItemDescriptions(nInfoItems)

```

EOSPAC User's Manual: Version 6.2

```

character*20 tableTypeLabel(nTables)
character*(EOS_MaxErrMsgLen) errorMessage
integer k

data infoItems /
&   EOS_Cmnt_Len,
&   EOS_Exchange_Coeff,
&   EOS_F_Convert_Factor,
&   EOS_Log_Val,
&   EOS_Material_ID,
&   EOS_Mean_Atomic_Mass,
&   EOS_Mean_Atomic_Num,
&   EOS_Modulus,
&   EOS_Normal_Density,
&   EOS_Table_Type,
&   EOS_X_Convert_Factor,
&   EOS_Y_Convert_Factor
&   /
data infoItemDescriptions /
&'The length in characters of the comments available for the specif
&ied data table',
&'The exchange coefficient',
&'The conversion factor corresponding to the dependent variable, F(
&x, y)',
&'Non-zero if the data table is in a log10 format',
&'The SESAME material identification number',
&'The mean atomic mass',
&'The mean atomic number',
&'The solid bulk modulus',
&'The normal density',
&'The type of data table. Corresponds to the parameters in APPENDIX
& B and APPENDIX C',
&'The conversion factor corresponding to the primary independent va
&riable, x',
&'The conversion factor corresponding to the secondary independent
&variable, y'
&/
data tableTypeLabel /
&   'EOS_Pt_DT',
&   'EOS_Dv_T',
&   'EOS_Ogb',
&   'EOS_Comment',
&   'EOS_Info'
&   /

c   EOS_Pt_DT, material 2140 works for Sesame table 301 (record type 1)
c   EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)
c   EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
c   EOS_Comment, material 2140 works for Sesame tables 101-199 (record
type 4)
c   EOS_Info, material 2140 works for Sesame table 201 (record type 5)
tableType(1) = EOS_Pt_DT
tableType(2) = EOS_Dv_T
tableType(3) = EOS_Ogb

```

```

        tableType(4) = EOS_Comment
        tableType(5) = EOS_Info

        numIndVars(1) = 2
        numIndVars(2) = 1
        numIndVars(3) = 0
        numIndVars(4) = 0
        numIndVars(5) = 0

        matID(1) = 2140
        matID(2) = 2140
        matID(3) = 12140
        matID(4) = 2140
        matID(5) = 2140

        errorCode = EOS_OK
        do 10 i=1, nTables
            tableHandle(i) = 0
10        continue

c
c        initialize table data objects
c
        call eos_CreateTables ( nTables, tableType, matID,
&                               tableHandle, errorCode)
        if (errorCode.NE.EOS_OK) then
            do 15 i=1, nTables
                tableHandleErrorCode = EOS_OK
                call eos_GetErrorCode
&                ( tableHandle(i), tableHandleErrorCode )
                call eos_GetErrorMessage
&                ( tableHandleErrorCode, errorMessage )
                call strLength(errorMessage, EOS_MaxErrMsgLen, k)
                write(*,998) 'eos_CreateTables ERROR ',tableHandleErrorCode,
&                ': ',errorMessage(1:k)
15        continue
            endif

c
c        set some options
c
        do 20 i=1, nTables
c        enable smoothing
            call eos_SetOption ( tableHandle(i), EOS_SMOOTH,
&                               EOS_NullVal, errorCode )
            if (errorCode.NE.EOS_OK) then
                call eos_GetErrorMessage ( errorCode, errorMessage )
                call strLength(errorMessage, EOS_MaxErrMsgLen, k)
                write(*,998) 'eos_SetOption ERROR ', errorCode,
&                ': ', errorMessage(1:k)
            endif
20        continue

c

```

```

c      load data into table data objects
c
      call eos_LoadTables ( nTables, tableHandle, errorCode)
      if (errorCode.NE.EOS_OK) then
        call eos_GetErrorMessage ( errorCode, errorMessage )
        call strLength(errorMessage, EOS_MaxErrMsgLen, k)
        write(*,998) 'eos_LoadTables ERROR ', errorCode, ': ',
&                errorMessage(1:k)
        do 25 i=1, nTables
          tableHandleErrorCode = EOS_OK
          call eos_GetErrorCode
&            ( tableHandle(i), tableHandleErrorCode )
          call eos_GetErrorMessage
&            ( tableHandleErrorCode, errorMessage )
          call strLength(errorMessage, EOS_MaxErrMsgLen, k)
          write(*,994) 'eos_LoadTables ERROR ', tableHandleErrorCode,
&                    ' (TH=', tableHandle(i), '): ',
&                    errorMessage(1:k)
25      continue
      endif

c
c      interpolate -- errors codes are intentionally produced
c
      X(1) = 3000.d0
      X(2) = 6000.d0
      X(3) = 8200.d0
      X(4) = 8300.d0

      Y(1) = 20000.0d0
      Y(2) = 620000.0d0
      Y(3) = 4000000.0d0
      Y(4) = 200000000.0d0

      do 30 i=1, nTables
        write(*,*) ' '
        write(*,997) '--- Interpolate using tableType ',
&                tableTypeLabel(i), ' ---'
&        call eos_Interpolate ( tableHandle(i), nXYPairs, X, Y, F,
&                dFx, dFy, errorCode)
&        write(*,997) tableTypeLabel(i), ' Interpolation Results:'
        if (errorCode.NE.EOS_OK) then
          call eos_GetErrorMessage ( errorCode, errorMessage )
          call strLength(errorMessage, EOS_MaxErrMsgLen, k)
          write(*,994) 'eos_Interpolate ERROR ', errorCode,
&                    ' (TH=', tableHandle(i), '): ',
&                    errorMessage(1:k)
&
&        else
          do 40 j=1, nXYPairs
            if (numIndVars(i).EQ.1) then
              write(*,996) j-1,X(j),F(j),dFx(j),errorCode
            endif
            if (numIndVars(i).EQ.2) then
              write(*,999) j-1,X(j),Y(j),F(j),dFx(j),dFy(j),errorCode

```

```

        endif
40      continue
      endif
30    continue

c
c    Retrieve all miscellaneous table info
c
    do 45 i=1, nTables
      write(*,*) ' '
      write(*,997) '--- Table information for tableType ',
&      tableTypeLabel(i), ', tableHandle=', tableHandle(i),
&      ' ---'
      do 50 j=1, nInfoItems
        call eos_GetTableInfo ( tableHandle(i), 1,
&      infoItems(j), infoVals(j), errorCode)
        if (errorCode.EQ.EOS_OK) then
          write(*,995) j, '. ', infoItemDescriptions(j), ': ',
&      infoVals(j)
        else if (errorCode.NE.EOS_INVALID_INFO_FLAG) then
c          Ignore EOS_INVALID_INFO_FLAG since not all infoItems are
currently
c          applicable to a specific tableHandle.
          call eos_GetErrorMessage ( errorCode, errorMessage )
          call strlen(errorMessage, EOS_MaxErrMsgLen, k)
          write(*,998) 'eos_LoadTables ERROR ', errorCode,
&      ': ', errorMessage(1:k)
        endif
50      continue
45    continue

c
c    Destroy all data objects
c
    call eos_DestroyAll (errorCode)
    if (errorCode.NE.EOS_OK) then
      do 35 i=1, nTables
        tableHandleErrorCode = EOS_OK
        call eos_GetErrorCode (
&      tableHandle(i), tableHandleErrorCode )
        call eos_GetErrorMessage (
&      tableHandleErrorCode, errorMessage )
        call strlen(errorMessage, EOS_MaxErrMsgLen, k)
        write(*,998) 'eos_DestroyAll ERROR ', tableHandleErrorCode,
&      ': ', errorMessage(1:k)
35      continue
    endif

994  format (a,i4,a,i1,2a)
995  format (i2,a,a,a,f13.6)
996  format ('    i=',i2,'    X =',1p13.6,
&      ', F =',1p13.6,'    dFx =',1p13.6,'    errorCode: ',i4)
997  format (a,:,a,:,2(a,:,i2))
998  format (a,i4,2a)

```

EOSPAC User's Manual: Version 6.2

```
999 format ('      i=',i2,'      X =',1pe13.6,' , Y =',1pe13.6,
&          ', F =',1pe13.6,' , dFx =',1pe13.6,' , dFy =',
&          1pe13.6,' , errorCode: ',i4)

      end

      subroutine strLength(str, length, trimmedLength)
      integer i, length, trimmedLength
      character*(*) str
      trimmedLength = 0
      do 5 i=length, 1, -1
          if (trimmedLength.EQ.0 .AND. str(i:i).NE.' ' .AND.
&          str(i:i).NE.char(0)) then
              trimmedLength = i
          endif
5      continue
      end
```

9.4 Fortran 90 Host Code Example

```
!*****
! Example F90 program
! -----
! Filetype: (HEADER)
!
! Copyright -- see file named COPYRIGHTNOTICE
!
!*****

!> @file
!! @ingroup examples
!! @brief This is a simple F90 example of how to use EOSPAC6 interface.
```

```
program TestF90

  use eos_Interface

  implicit none

  integer(EOS_INTEGER),parameter :: nTables = 5
  integer(EOS_INTEGER),parameter :: nXYPairs = 4
  integer(EOS_INTEGER),parameter :: nInfoItems = 12

  integer(EOS_INTEGER) :: i, j
  real(EOS_REAL) :: X(nXYPairs), Y(nXYPairs), F(nXYPairs), dFx(nXYPairs),
dFy(nXYPairs)
  integer(EOS_INTEGER) :: tableType(nTables), numIndVars(nTables)
  integer(EOS_INTEGER) :: matID(nTables)
  integer(EOS_INTEGER) :: tableHandle(nTables)
  integer(EOS_INTEGER) :: errorCode
```

```

integer(EOS_INTEGER) :: tableHandleErrorCode
real(EOS_REAL) :: infoVals(nInfoItems)
integer(EOS_INTEGER) :: infoItems(nInfoItems) = (/ &
    EOS_Cmnt_Len, &
    EOS_Exchange_Coeff, &
    EOS_F_Convert_Factor, &
    EOS_Log_Val, &
    EOS_Material_ID, &
    EOS_Mean_Atomic_Mass, &
    EOS_Mean_Atomic_Num, &
    EOS_Modulus, &
    EOS_Normal_Density, &
    EOS_Table_Type, &
    EOS_X_Convert_Factor, &
    EOS_Y_Convert_Factor &
    /)
character(82) :: infoItemDescriptions(nInfoItems) = (/ &
    'The length in characters of the comments available for the
specified data table ', &
    'The exchange coefficient
', &
    'The conversion factor corresponding to the dependent variable,
F(x,y) ', &
    'Non-zero if the data table is in a log10 format
', &
    'The SESAME material identification number
', &
    'The mean atomic mass
', &
    'The mean atomic number
', &
    'The solid bulk modulus
', &
    'The normal density
', &
    'The type of data table. Corresponds to the parameters in APPENDIX
B and APPENDIX C', &
    'The conversion factor corresponding to the primary independent
variable, x ', &
    'The conversion factor corresponding to the secondary independent
variable, y ' &
    /)
character(11) :: tableTypeLabel(nTables) = (/ &
    'EOS_Pt_DT ', &
    'EOS_Dv_T ', &
    'EOS_Ogb ', &
    'EOS_Comment', &
    'EOS_Info ' &
    /)
character(EOS_MaxErrMsgLen) :: errorMessage

!     EOS_Pt_DT, material 2140 works for Sesame table 301 (record type
1)
!     EOS_Dv_T, material 2140 works for Sesame table 401 (record type 2)

```

EOSPAC User's Manual: Version 6.2

```
!      EOS_Ogb, material 12140 works for Sesame table 501 (record type 3)
!      EOS_Comment, material 2140 works for Sesame tables 101-199 (record
type 4)
!      EOS_Info, material 2140 works for Sesame table 201 (record type 5)
tableType(1) = EOS_Pt_DT
tableType(2) = EOS_Dv_T
tableType(3) = EOS_Ogb
tableType(4) = EOS_Comment
tableType(5) = EOS_Info

numIndVars(1) = 2
numIndVars(2) = 1
numIndVars(3) = 0
numIndVars(4) = 0
numIndVars(5) = 0

matID(1) = 2140
matID(2) = 2140
matID(3) = 12140
matID(4) = 2140
matID(5) = 2140

errorCode = EOS_OK
do i=1, nTables
  tableHandle(i) = 0
enddo

!
!      initialize table data objects
!
call eos_CreateTables ( nTables, tableType, matID, tableHandle,
errorCode)
if (errorCode.NE.EOS_OK) then
  do i=1, nTables
    tableHandleErrorCode = EOS_OK
    call eos_GetErrorCode ( tableHandle(i), tableHandleErrorCode )
    call eos_GetErrorMessage ( tableHandleErrorCode, errorMessage )
    write(*,998) 'eos_CreateTables ERROR ', tableHandleErrorCode, ':
', &
                errorMessage(1:(len_trim(errorMessage)-1))
  enddo
endif

!
!      set some options
!
do i=1, nTables
  !      enable smoothing
  call eos_SetOption ( tableHandle(i), EOS_SMOOTH, EOS_NullVal,
errorCode )
  if (errorCode.NE.EOS_OK) then
    call eos_GetErrorMessage ( errorCode, errorMessage )
    write(*,998) 'eos_SetOption ERROR ', errorCode, ': ', &
                errorMessage(1:(len_trim(errorMessage)-1))
```

```

        endif
    enddo

    !
    !     load data into table data objects
    !
    call eos_LoadTables ( nTables, tableHandle, errorCode)
    if (errorCode.NE.EOS_OK) then
        call eos_GetErrorMessage ( errorCode, errorMessage )
        write(*,998) 'eos_LoadTables ERROR ', errorCode, ': ', &
            errorMessage(1:(len_trim(errorMessage)-1))
        do i=1, nTables
            tableHandleErrorCode = EOS_OK
            call eos_GetErrorCode ( tableHandle(i), tableHandleErrorCode )
            call eos_GetErrorMessage ( tableHandleErrorCode, errorMessage )
            write(*,994) 'eos_LoadTables ERROR ', tableHandleErrorCode, '
(TH=', &
                tableHandle(i), '): ', &
                errorMessage(1:(len_trim(errorMessage)-1))
        enddo
    endif

    !
    !     interpolate -- errors codes are intentionally produced
    !
    X(1) = 3000._EOS_REAL
    X(2) = 6000._EOS_REAL
    X(3) = 8200._EOS_REAL
    X(4) = 8300._EOS_REAL

    Y(1) = 20000.0_EOS_REAL
    Y(2) = 620000.0_EOS_REAL
    Y(3) = 4000000.0_EOS_REAL
    Y(4) = 200000000.0_EOS_REAL

    do i=1, nTables
        write(*,*) ' '
        write(*,997) '--- Interpolate using tableType ', tableTypeLabel(i), '
---'
        call eos_Interpolate ( tableHandle(i), nXYPairs, X, Y, F, dFx, dFy,
            errorCode)
        write(*,997) tableTypeLabel(i), ' Interpolation Results:'
        if (errorCode.NE.EOS_OK) then
            call eos_GetErrorMessage ( errorCode, errorMessage )
            write(*,994) 'eos_Interpolate ERROR ', errorCode, ' (TH=', &
                tableHandle(i), '): ', &
                errorMessage(1:(len_trim(errorMessage)-1))
        else
            do j=1, nXYPairs
                if (numIndVars(i).EQ.1) then
                    write(*,996) j-1,X(j),F(j),dFx(j),errorCode
                endif
                if (numIndVars(i).EQ.2) then
                    write(*,999) j-1,X(j),Y(j),F(j),dFx(j),dFy(j),errorCode
                endif
            enddo
        endif
    enddo

```

EOSPAC User's Manual: Version 6.2

```

        endif
    enddo
endif
enddo

!
!   Retrieve all miscellaneous table info
!
do i=1, nTables
    write(*,*) ' '
    write(*,997) '--- Table information for tableType ',
tableTypeLabel(i), &
                ', tableHandle=', tableHandle(i), ' ---'
    do j=1, nInfoItems
        call eos_GetTableInfo ( tableHandle(i), 1_EOS_INTEGER,
infoItems(j), &
                            infoVals(j), errorCode )
        if (errorCode.EQ.EOS_OK) then
            write(*,995) j, ' ', infoItemDescriptions(j), ': ', infoVals(j)
        else if (errorCode.NE.EOS_INVALID_INFO_FLAG) then
            ! Ignore EOS_INVALID_INFO_FLAG since not all infoItems are
currently
            ! applicable to a specific tableHandle.
            call eos_GetErrorMessage ( errorCode, errorMessage )
            write(*,998) 'eos_GetTableInfo ERROR ', errorCode, ': ', &
                errorMessage(1:(len_trim(errorMessage)-1))
        endif
    enddo
enddo

!
!   Destroy all data objects
!
call eos_DestroyAll (errorCode)
if (errorCode.NE.EOS_OK) then
    do i=1, nTables
        tableHandleErrorCode = EOS_OK
        call eos_GetErrorCode ( tableHandle(i), tableHandleErrorCode )
        call eos_GetErrorMessage ( tableHandleErrorCode, errorMessage )
        write(*,998) 'eos_DestroyAll ERROR ', tableHandleErrorCode, ': ',
&
                errorMessage(1:(len_trim(errorMessage)-1))
    enddo
endif

994 format (a,i4,a,i1,2a)
995 format (i2,a,a,a,f13.6)
996 format ('   i=',i2,'   X =',1pe13.6, &
           ', F =',1pe13.6,', dFx =',1pe13.6,', errorCode: ',i4)
997 format (a,:,a,:,2(a,:,i2))
998 format (a,i4,2a)
999 format ('   i=',i2,'   X =',1pe13.6,', Y =',1pe13.6, &
           ', F =',1pe13.6,', dFx =',1pe13.6,', dFy =', &
           1pe13.6,', errorCode: ',i4)

```

```
end program TestF90
```

10 TECHNICAL SUPPORT INFORMATION

The X Division Data Team provides online documentation and references related to EOSPAC at the following URL on both the open and secure networks:

<https://xweb.lanl.gov/PROJECTS/DATA/eos/>

Technical support, bug reports and feature requests for EOSPAC version 6 can be obtained or submitted by contacting the Data Team via the EOSPAC version 6 mailing list, which is available on both the open and secure networks:

help-eospac6@sf4.lanl.gov

The X Division Data Team member(s) responsible for EOSPAC are listed as follows:

David A. Pimentel

Los Alamos National Laboratory

MS F663

HPC-1, TA-3-1400, Rm. 4116

Los Alamos, New Mexico 87545

Email: davidp@lanl.gov

Phone: (505) 665-1255

11 ACKNOWLEDGEMENTS

Bill Archer, formerly of CCN-12, deserves many thanks for his large contributions to the initial planning and documentation of the EOSPAC 6 development project.

Olga Chotinun, formerly of HPC-1, was instrumental in the development of the software package.

Angela Herring, formerly of X-1-NAD, contributed many significant fixes and enhancements, and she deserves many thanks.

12 REFERENCES

1. S. P. Lyon and J. D. Johnson, *SESAME: The Los Alamos National Laboratory Equation Of State Database*, Los Alamos National Laboratory report [LA-UR-92-3407](#) (1992).
2. C. W. Cranfill, *The EOSPAC utility package for accessing the Sesame data library*, Los Alamos National Laboratory memorandum X7-87-U53 (March 31, 1987).
3. D. George, *OpenSesame*, Los Alamos National Laboratory software LA-CC-03-087 (October 21, 2003).
4. C. W. Cranfill and R. More, *IONEOS: A Fast, Analytic, Ion Equation-of-State Routine*, Los Alamos National Laboratory report LA-7313-MS (October 1978).
5. ISO/IEC Std 9945-1:1990; IEEE Std 1003.1-1990, *Information Technology – Portable Operating System Interface (POSIX), Part 1: System Application Program Interface (API) (C language)*, September 1990.
6. Donald Lewine, *POSIX Programmer's Guide*, O'Reilly & Associates, Inc., Sebastopol, CA, 1994.
7. C. W. Cranfill, *EOS of a Material Mixture in Pressure Equilibrium*, Los Alamos National Laboratory report [LA-13661](#) (January 2000).
8. D. A. Pimentel, *EOSPAC 5 Users Document: Version 5.30*, 3-May-2001, <http://laurel.lanl.gov/PROJECTS/DATA/eos/UsersDocument/HTML/EOSPAC.html>.
9. C. W. Cranfill, *MIXPAC: A Subroutine Package For Calculating Equations Of State For Equilibrium Mixtures Of Materials*, Los Alamos National Laboratory report [LA-9861-M](#) (August 1983).
10. C. W. Cranfill, *The EOSPAC Utility Package for Accessing the SESAME Data Library*, Los Alamos National Laboratory internal memorandum X7-87-U53 (March 31, 1987).
11. Brian J. Gough, foreword by Richard M. Stallman, *An Introduction to GCC - for the GNU compilers gcc and g++*, Paperback (6"x9"), 124 pages, ISBN 0954161793, <http://www.network-theory.co.uk/docs/gccintro/index.html>.

APPENDIX A. TABLE TYPE IDENTIFIER NAME MNEMONIC CONVENTIONS

Below is an alphabetized list of mnemonics used to create the EOSPAC table type identifier names that are defined in both APPENDIX B and APPENDIX C. These mnemonics are combined as follows to create the aforementioned identifier names:

EOS_#_@\$

where # is the mnemonic of the dependent function, $F(x,y)$

@ is the mnemonic of the primary independent variable, x

\$ is the mnemonic of the secondary independent variable, y.

Mnemonic	Description
Ac	Specific-Helmholtz-Free-Energy Cold Curve
Ae	Electron Specific-Helmholtz-Free-Energy
Af	Freeze Specific-Helmholtz-Free-Energy
Aic	Ion Specific-Helmholtz-Free-Energy plus Cold Curve Specific-Helmholtz-Free-Energy
Aiz	Ion Specific-Helmholtz-Free-Energy Including Zero Point
Als	Liquid or Solid Specific-Helmholtz-Free-Energy
Am	Melt Specific-Helmholtz-Free-Energy
At	Total Specific-Helmholtz-Free-Energy
Av	Vapor Specific-Helmholtz-Free-Energy
B	Thermoelectric Coefficient
Comment	Descriptive Comments
D	Density
Dls	Liquid or Solid Density on coexistence line
Dv	Vapor Density on coexistence line
Gs	Shear Modulus
Info	Atomic Number, Atomic Mass, Normal Density, Solid Bulk Modulus, Exchange Coefficient
Kc	Electron Conductive Opacity (Conductivity Model)
Kec	Electrical Conductivity
Keo	Electron Conductive Opacity (Opacity Model)
Kp	Planck Mean Opacity

Mnemonic	Description
Kr	Rosseland Mean Opacity
Ktc	Thermal Conductivity
M	Mass fraction
NullTable	null table
Ogb	Calculated versus Interpolated Opacity Grid Boundary
Pc	Pressure Cold Curve
Pe	Electron Pressure
Pf	Freeze Pressure
Pic	Ion Pressure plus Cold Curve Pressure
Piz	Ion Pressure Including Zero Point
Pm	Melt Pressure
Pt	Total Pressure
Pv	Vapor Pressure
Se	Electron Specific-Entropy
Sic	Ion Pressure plus Cold Curve Specific-Entropy
Siz	Ion Pressure Including Zero Specific-Entropy
St	Total Specific-Entropy
T	Temperature
Tf	Freeze Temperature
Tm	Melt Temperature
Uc	Specific-Internal-Energy Cold Curve
Ue	Electron Specific-Internal-Energy
Uf	Freeze Specific-Internal-Energy
Uic	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy
Uiz	Ion Specific-Internal-Energy Including Zero Point
Uls	Liquid or Solid Specific-Internal-Energy
Um	Melt Specific-Internal-Energy
Ut	Total Specific-Internal-Energy
Uv	Vapor Specific-Internal-Energy
V	Specific Volume
Zfc	Mean Ion Charge (Conductivity Model)

EOSPAC User's Manual: Version 6.2

Mnemonic	Description
Zfo	Mean Ion Charge (Opacity Model)

APPENDIX B. TABLE TYPE DEFINITIONS GROUPED BY CATEGORY AND SORTED BY IDENTIFIER NAME

Below is a list of defined constants⁸ corresponding to all of the data table types available within EOSPAC. These defined constants have been grouped into several data categories, alphabetized according to the defined constant names, and cross-referenced to the applicable EOSPAC 5 defined constants. The constant names have been created using the mnemonics defined in APPENDIX A. The data categories are listed below and referenced to pages within this appendix.

<u>Category</u>	<u>Description</u>	<u>Page</u>
CATEGORY 1	UNRELATED TO SESAME DATA	2
CATEGORY 2	GENERAL INFORMATION FOUND IN SESAME'S 100- AND 200-SERIES TABLES	3
CATEGORY 3	TOTAL EOS IN SESAME'S 301 TABLES	4
CATEGORY 4	ION+COLD EOS IN SESAME'S 303 TABLES	7
CATEGORY 5	ELECTRON EOS IN SESAME'S 304 TABLES	10
CATEGORY 6	ION EOS IN SESAME'S 305 TABLES	12
CATEGORY 7	COLD CURVE EOS IN SESAME'S 306 TABLES	15
CATEGORY 8	MASS FRACTION EOS IN SESAME'S 321 TABLES	16
CATEGORY 9	VAPORIZATION DATA IN SESAME'S 401 TABLES	17
CATEGORY 10	MELT DATA IN SESAME'S 411 AND 412 TABLES	22
CATEGORY 12	OPACITY DATA IN SESAME'S 500-SERIES TABLES	26
CATEGORY 13	CONDUCTIVITY DATA IN SESAME'S 600-SERIES TABLES	27

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code. The EOSPAC 6 Constants are color coded as follows:

- **Red Text** indicates the table is inverted with respect to the first independent variable.
- **Green Text** indicates the table is inverted with respect to the second independent variable.
- **Blue Text** indicates the table is a combination of two other tables.

⁸ This is the proposed final list as of this writing (1/5/2011 12:37 PM).

Category 1 Unrelated to Sesame data

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_NullTable	null table	n/a

Category 2 General information found in SESAME's 100- and 200-series tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Comment	Descriptive Comments	101-199
EOS_Info	Atomic Number, Atomic Mass, Normal Density, Solid Bulk Modulus, Exchange Coefficient	201

Category 3 Total EOS in SESAME's 301 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_At_DPt	Total Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Total Pressure (GPa)-dependent)	301
EOS_At_DSt	Total Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Total Specific-Entropy (MJ/kg/K)-dependent)	301
EOS_At_DT	Total Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	301
EOS_At_DUt	Total Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_D_PtT	Density (Mg/m ³) (Total Pressure (GPa)- and Temperature (K)-dependent)	301
EOS_Pt_DAt	Total Pressure (GPa) (Density (Mg/m ³)- and Total Specific-Free-Energy (MJ/kg)-dependent)	301
EOS_Pt_DSt	Total Pressure (GPa) (Density (Mg/m ³)- and Total Specific-Entropy (MJ/kg/K)-dependent)	301
EOS_Pt_DT	Total Pressure (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	301
EOS_Pt_DUt	Total Pressure (GPa) (Density (Mg/m ³)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_St_DAt	Total Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Total Specific-Free-Energy (MJ/kg)-dependent)	301
EOS_St_DPt	Total Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Total Pressure (GPa)-dependent)	301
EOS_St_DT	Total Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Temperature (K)-dependent)	301
EOS_St_DUt	Total Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_T_DAt	Temperature (K) (Density (Mg/m ³)- and Total Specific-Free-Energy (MJ/kg)-dependent)	301
EOS_T_DPt	Temperature (K) (Density (Mg/m ³)- and Total Pressure (GPa)-dependent)	301
EOS_T_DSt	Temperature (K) (Density (Mg/m ³)- and Total Specific-Entropy (MJ/kg/K)-dependent)	301
EOS_T_DUt	Temperature (K) (Density (Mg/m ³)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Ut_DAt	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Total Specific-Free-Energy (MJ/kg)-dependent)	301
EOS_Ut_DPt	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Total Pressure (GPa)-dependent)	301
EOS_Ut_DSt	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Total Specific-Entropy (MJ/kg/K)-dependent)	301

EOSPAC 6 Constant	Description	SESAME Table(s)
	(MJ/kg/K)-dependent	
EOS_Ut_DT	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	301
EOS_Ut_PtT	Total Specific-Internal-Energy (MJ/kg) (Total Pressure (GPa)- and Temperature (K)-dependent) NOTE: This data table type is only usable in conjunction with the related EOS_PT_SMOOTHING and EOS_USE_CUSTOM_INTERP options.	301
EOS_V_PtT	Specific-Volume (m ³ /Mg) (Total Pressure (GPa)- and Temperature (K)-dependent)	301

Category 4 Ion+Cold EOS in SESAME's 303 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Aic_DPic	Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Aic_DSic	Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K)-dependent)	303
EOS_Aic_DT	Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	303
EOS_Aic_DUic	Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Pic_DAic	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m ³)- and Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg)-dependent)	303
EOS_Pic_DSic	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K)-dependent)	303
EOS_Pic_DT	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	303
EOS_Pic_DUic	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m ³)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Sic_DAic	Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg)-dependent)	303
EOS_Sic_DPic	Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Sic_DT	Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Temperature (K)-dependent)	303
EOS_Sic_DUic	Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_T_DAic	Temperature (K) (Density (Mg/m ³)- and Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg)-dependent)	303
EOS_T_DPic	Temperature (K) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_T_DSic	Temperature (K) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K)-dependent)	303
EOS_T_DUic	Temperature (K) (Density (Mg/m ³)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Uic_DAic	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Specific-Free-Energy plus Cold Curve Specific-Free-Energy (MJ/kg)-dependent)	303
EOS_Uic_DPic	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_Uic_DSic	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Specific-Entropy (MJ/kg/K)-dependent)	303
EOS_Uic_DT	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	303

Category 5 Electron EOS in SESAME's 304 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Ae_DPe	Electron Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Pressure (GPa)-dependent)	304
EOS_Ae_DSe	Electron Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Specific-Entropy (MJ/kg/K)-dependent)	304
EOS_Ae_DT	Electron Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	304
EOS_Ae_DUe	Electron Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Pe_D Ae	Electron Pressure (GPa) (Density (Mg/m ³)- and Electron Specific-Free-Energy (MJ/kg)-dependent)	304
EOS_Pe_DSe	Electron Pressure (GPa) (Density (Mg/m ³)- and Electron Specific-Entropy (MJ/kg/K)-dependent)	304
EOS_Pe_DT	Electron Pressure (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	304
EOS_Pe_DUe	Electron Pressure (GPa) (Density (Mg/m ³)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Se_D Ae	Electron Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Electron Specific-Free-Energy (MJ/kg)-dependent)	304
EOS_Se_DPe	Electron Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Electron Pressure (GPa)-	304

EOSPAC 6 Constant	Description	SESAME Table(s)
	dependent)	
EOS_Se_DT	Electron Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Temperature (K)-dependent)	304
EOS_Se_DUe	Electron Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_T_DAe	Temperature (K) (Density (Mg/m ³)- and Electron Specific-Free-Energy (MJ/kg)-dependent)	304
EOS_T_DPe	Temperature (K) (Density (Mg/m ³)- and Electron Pressure (GPa)-dependent)	304
EOS_T_DSe	Temperature (K) (Density (Mg/m ³)- and Electron Specific-Entropy (MJ/kg/K)-dependent)	304
EOS_T_DUe	Temperature (K) (Density (Mg/m ³)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_Ue_DAe	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Specific-Free-Energy (MJ/kg)-dependent)	304
EOS_Ue_DPe	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Pressure (GPa)-dependent)	304
EOS_Ue_DSe	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Specific-Entropy (MJ/kg/K)-dependent)	304
EOS_Ue_DT	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	304

Category 6 Ion EOS in SESAME's 305 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Aiz_DPiz	Ion Specific-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Aiz_DSiz	Ion Specific-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Ion Pressure Including Zero Specific-Entropy (MJ/kg/K)-dependent)	305
EOS_Aiz_DT	Ion Specific-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	305
EOS_Aiz_DUiz	Ion Specific-Free-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Piz_DAiz	Ion Pressure Including Zero Point (GPa) (Density (Mg/m ³)- and Ion Specific-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Piz_DSiz	Ion Pressure Including Zero Point (GPa) (Density (Mg/m ³)- and Ion Pressure Including Zero Specific-Entropy (MJ/kg/K)-dependent)	305
EOS_Piz_DT	Ion Pressure Including Zero Point (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	305
EOS_Piz_DUiz	Ion Pressure Including Zero Point (GPa) (Density (Mg/m ³)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Siz_DAiz	Ion Pressure Including Zero Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Ion Specific-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Siz_DPiz	Ion Pressure Including Zero Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Siz_DT	Ion Pressure Including Zero Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Temperature (K)-dependent)	305
EOS_Siz_DUiz	Ion Pressure Including Zero Specific-Entropy (MJ/kg/K) (Density (Mg/m ³)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_T_DAiz	Temperature (K) (Density (Mg/m ³)- and Ion Specific-Free-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_T_DPiz	Temperature (K) (Density (Mg/m ³)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_T_DSiz	Temperature (K) (Density (Mg/m ³)- and Ion Pressure Including Zero Specific-Entropy (MJ/kg/K)-dependent)	305
EOS_T_DUiz	Temperature (K) (Density (Mg/m ³)- and Ion Specific-Internal-Energy Including Zero Point (MJ/kg)-dependent)	305
EOS_Uiz_DAiz	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Ion Specific-Free-Energy Including Zero Point (MJ/kg)-dependent)	305

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Uiz_DPiz	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Ion Pressure Including Zero Point (GPa)-dependent)	305
EOS_Uiz_DSiz	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Ion Pressure Including Zero Specific-Entropy (MJ/kg/K)-dependent)	305
EOS_Uiz_DT	Ion Specific-Internal-Energy Including Zero Point (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	305

Category 7 Cold curve EOS in SESAME's 306 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Ac_D	Specific-Free-Energy Cold Curve (MJ/kg) (Density (Mg/m ³)-dependent)	306
EOS_Pc_D	Pressure Cold Curve (GPa) (Density (Mg/m ³)-dependent)	306
EOS_Uc_D	Specific-Internal-Energy Cold Curve (MJ/kg) (Density (Mg/m ³)-dependent)	306

Category 8 Mass Fraction EOS in SESAME's 321 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_M_DT	Mass Fraction (Density- and Temperature- dependent) See section 8.4 for further discussion about how to use this data type.	321

Category 9 Vaporization data in SESAME's 401 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Als_Av	Liquid or Solid Specific-Free-Energy (MJ/kg) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Als_Dls	Liquid or Solid Specific-Free-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Als_Dv	Liquid or Solid Specific-Free-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m ³)- dependent)	401
EOS_Als_Pv	Liquid or Solid Specific-Free-Energy (MJ/kg) (Vapor Pressure (GPa)-dependent)	401
EOS_Als_T	Liquid or Solid Specific-Free-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Als_Uls	Liquid or Solid Specific-Free-Energy (MJ/kg) (Liquid or Solid Specific-Internal-Energy (MJ/kg)- dependent)	401
EOS_Als_Uv	Liquid or Solid Specific-Free-Energy (MJ/kg) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Av_Dls	Vapor Specific-Free-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Av_Dv	Vapor Specific-Free-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m ³)- dependent)	401
EOS_Av_Gls	Vapor Specific-Free-Energy (MJ/kg) (Liquid or Solid Specific-Free-Energy (MJ/kg)- dependent)	401
EOS_Av_Pv	Vapor Specific-Free-Energy (MJ/kg)	401

EOSPAC 6 Constant	Description	SESAME Table(s)
	(Vapor Pressure (GPa)-dependent)	
EOS_Av_T	Vapor Specific-Free-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Av_Uls	Vapor Specific-Free-Energy (MJ/kg) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Av_Uv	Vapor Specific-Free-Energy (MJ/kg) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dls_Als	Liquid or Solid Density on coexistence line (Mg/m ³) (Liquid or Solid Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Dls_Av	Liquid or Solid Density on coexistence line (Mg/m ³) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Dls_Dv	Liquid or Solid Density on coexistence line (Mg/m ³) (Vapor Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Dls_Pv	Liquid or Solid Density on coexistence line (Mg/m ³) (Vapor Pressure (GPa)-dependent)	401
EOS_Dls_T	Liquid or Solid Density on coexistence line (Mg/m ³) (Temperature (K)-dependent)	401
EOS_Dls_Uls	Liquid or Solid Density on coexistence line (Mg/m ³) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dls_Uv	Liquid or Solid Density on coexistence line (Mg/m ³) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dv_Als	Vapor Density on coexistence line (Mg/m ³) (Liquid or Solid Specific-Free-Energy (MJ/kg)-dependent)	401

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Dv_Av	Vapor Density on coexistence line (Mg/m ³) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Dv_Dls	Vapor Density on coexistence line (Mg/m ³) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Dv_Pv	Vapor Density on coexistence line (Mg/m ³) (Vapor Pressure (GPa)-dependent)	401
EOS_Dv_T	Vapor Density on coexistence line (Mg/m ³) (Temperature (K)-dependent)	401
EOS_Dv_Uls	Vapor Density on coexistence line (Mg/m ³) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Dv_Uv	Vapor Density on coexistence line (Mg/m ³) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Pv_Als	Vapor Pressure (GPa) (Liquid or Solid Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Pv_Av	Vapor Pressure (GPa) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Pv_Dls	Vapor Pressure (GPa) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Pv_Dv	Vapor Pressure (GPa) (Vapor Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Pv_T	Vapor Pressure (GPa) (Temperature (K)-dependent)	401
EOS_Pv_Uls	Vapor Pressure (GPa) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Pv_Uv	Vapor Pressure (GPa) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_T_Als	Temperature (K) (Liquid or Solid Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_T_Av	Temperature (K) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_T_Dls	Temperature (K) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_T_Dv	Temperature (K) (Vapor Density on coexistence line (Mg/m ³)-dependent)	401
EOS_T_Pv	Temperature (K) (Vapor Pressure (GPa)-dependent)	401
EOS_T_Uls	Temperature (K) (Liquid or Solid Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_T_Uv	Temperature (K) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Uls_Als	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Liquid or Solid Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Uls_Av	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Uls_Dls	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Uls_Dv	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m ³)-dependent)	401

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Uls_Pv	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Vapor Pressure (GPa)-dependent)	401
EOS_Uls_T	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Uls_Uv	Liquid or Solid Specific-Internal-Energy (MJ/kg) (Vapor Specific-Internal-Energy (MJ/kg)-dependent)	401
EOS_Uv_Als	Vapor Specific-Internal-Energy (MJ/kg) (Liquid or Solid Specific-Free-Energy (MJ/kg)- dependent)	401
EOS_Uv_Av	Vapor Specific-Internal-Energy (MJ/kg) (Vapor Specific-Free-Energy (MJ/kg)-dependent)	401
EOS_Uv_Dls	Vapor Specific-Internal-Energy (MJ/kg) (Liquid or Solid Density on coexistence line (Mg/m ³)-dependent)	401
EOS_Uv_Dv	Vapor Specific-Internal-Energy (MJ/kg) (Vapor Density on coexistence line (Mg/m ³)- dependent)	401
EOS_Uv_Pv	Vapor Specific-Internal-Energy (MJ/kg) (Vapor Pressure (GPa)-dependent)	401
EOS_Uv_T	Vapor Specific-Internal-Energy (MJ/kg) (Temperature (K)-dependent)	401
EOS_Uv_Uls	Vapor Specific-Internal-Energy (MJ/kg) (Liquid or Solid Specific-Internal-Energy (MJ/kg)- dependent)	401

Category 10 Melt data in SESAME's 411 and 412 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Af_D	Freeze Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)-dependent)	412
EOS_Af_Pf	Freeze Specific-Free-Energy (MJ/kg) (Freeze Pressure (GPa)-dependent)	412
EOS_Af_Tf	Freeze Specific-Free-Energy (MJ/kg) (Freeze Temperature (K)-dependent)	412
EOS_Af_Uf	Freeze Specific-Free-Energy (MJ/kg) (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_Am_D	Melt Specific-Free-Energy (MJ/kg) (Density (Mg/m ³)-dependent)	411
EOS_Am_Pm	Melt Specific-Free-Energy (MJ/kg) (Melt Pressure (GPa)-dependent)	411
EOS_Am_Tm	Melt Specific-Free-Energy (MJ/kg) (Melt Temperature (K)-dependent)	411
EOS_Am_Um	Melt Specific-Free-Energy (MJ/kg) (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_D_Af	Density (Mg/m ³) (Freeze Specific-Free-Energy (MJ/kg)-dependent)	412
EOS_D_Am	Density (Mg/m ³) (Melt Specific-Free-Energy (MJ/kg)-dependent)	411
EOS_D_Pf	Density (Mg/m ³) (Freeze Pressure (GPa)-dependent)	412
EOS_D_Pm	Density (Mg/m ³) (Melt Pressure (GPa)-dependent)	411
EOS_D_Tf	Density (Mg/m ³) (Freeze Temperature (K)-dependent)	412

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_D_Tm	Density (Mg/m ³) (Melt Temperature (K)-dependent)	411
EOS_D_Uf	Density (Mg/m ³) (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_D_Um	Density (Mg/m ³) (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_Pf_Af	Freeze Pressure (GPa) (Freeze Specific-Free-Energy (MJ/kg)-dependent)	412
EOS_Pf_D	Freeze Pressure (GPa) (Density (Mg/m ³)-dependent)	412
EOS_Pf_Tf	Freeze Pressure (GPa) (Freeze Temperature (K)-dependent)	412
EOS_Pf_Uf	Freeze Pressure (GPa) (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_Pm_Am	Melt Pressure (GPa) (Melt Specific-Free-Energy (MJ/kg)-dependent)	411
EOS_Pm_D	Melt Pressure (GPa) (Density (Mg/m ³)-dependent)	411
EOS_Pm_Tm	Melt Pressure (GPa) (Melt Temperature (K)-dependent)	411
EOS_Pm_Um	Melt Pressure (GPa) (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_Tf_Af	Freeze Temperature (K) (Freeze Specific-Free-Energy (MJ/kg)-dependent)	412
EOS_Tf_D	Freeze Temperature (K) (Density (Mg/m ³)-dependent)	412
EOS_Tf_Pf	Freeze Temperature (K) (Freeze Pressure (GPa)-dependent)	412

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Tf_Uf	Freeze Temperature (K) (Freeze Specific-Internal-Energy (MJ/kg)-dependent)	412
EOS_Tm_Am	Melt Temperature (K) (Melt Specific-Free-Energy (MJ/kg)-dependent)	411
EOS_Tm_D	Melt Temperature (K) (Density (Mg/m ³)-dependent)	411
EOS_Tm_Pm	Melt Temperature (K) (Melt Pressure (GPa)-dependent)	411
EOS_Tm_Um	Melt Temperature (K) (Melt Specific-Internal-Energy (MJ/kg)-dependent)	411
EOS_Uf_Af	Freeze Specific-Internal-Energy (MJ/kg) (Freeze Specific-Free-Energy (MJ/kg)-dependent)	412
EOS_Uf_D	Freeze Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)-dependent)	412
EOS_Uf_Pf	Freeze Specific-Internal-Energy (MJ/kg) (Freeze Pressure (GPa)-dependent)	412
EOS_Uf_Tf	Freeze Specific-Internal-Energy (MJ/kg) (Freeze Temperature (K)-dependent)	412
EOS_Um_Am	Melt Specific-Internal-Energy (MJ/kg) (Melt Specific-Free-Energy (MJ/kg)-dependent)	411
EOS_Um_D	Melt Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)-dependent)	411
EOS_Um_Pm	Melt Specific-Internal-Energy (MJ/kg) (Melt Pressure (GPa)-dependent)	411
EOS_Um_Tm	Melt Specific-Internal-Energy (MJ/kg) (Melt Temperature (K)-dependent)	411

Category 11 Shear Modulus data in SESAME's 431 tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_D_Gs	Density (Mg/m ³) (Shear Modulus (Gpa)-dependent)	431
EOS_Gs_D	Shear Modulus (Gpa) (Density (Mg/m ³)-dependent)	431

Category 12 Opacity data in SESAME's 500-series tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_Keo_DT	Electron Conductive Opacity (Opacity Model) (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)- dependent)	503
EOS_Kp_DT	Planck Mean Opacity (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)- dependent)	505
EOS_Kr_DT	Rosseland Mean Opacity (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)- dependent)	502
EOS_Ogb	Calculated versus Interpolated Opacity Grid Boundary	501
EOS_Zfo_DT	Mean Ion Charge (Opacity Model) (free electrons per atom) (Density (Mg/m ³)- and Temperature (eV)- dependent)	504

Category 13 Conductivity data in SESAME's 600-series tables

EOSPAC 6 Constant	Description	SESAME Table(s)
EOS_B_DT	Thermoelectric Coefficient (1/cm ² /s) (Density (Mg/m ³)- and Temperature (eV)- dependent)	604
EOS_Kc_DT	Electron Conductive Opacity (Conductivity Model) (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)- dependent)	605
EOS_Kec_DT	Electrical Conductivity (1/s) (Density (Mg/m ³)- and Temperature (eV)- dependent)	602
EOS_Ktc_DT	Thermal Conductivity (1/cm/s) (Density (Mg/m ³)- and Temperature (eV)- dependent)	603
EOS_Zfc_DT	Mean Ion Charge (Conductivity Model) (free electrons per atom) (Density (Mg/m ³)- and Temperature (eV)- dependent)	601

APPENDIX C. TABLE TYPE DEFINITIONS CROSS REFERENCED TO THOSE OF EOSPAC VERSION 5

Below are tables of defined constants⁹ corresponding to all of the data table types available within EOSPAC version 5.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code. The EOSPAC 6 Constants are color coded as follows:

- **Red Text** indicates the table is inverted with respect to the first independent variable.
- **Green Text** indicates the table is inverted with respect to the second independent variable.
- **Blue Text** indicates the table is a combination of two other tables.
- « indicates the table is compatible with the eos_Mix routine.

EOSPAC 6 Constant		EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_B_DT	«	ES4_THERME	Thermoelectric Coefficient (1/cm ² /s) (Density (Mg/m ³)- and Temperature (eV)-dependent)	604
EOS_D_PtT		ES4_DPTTOT	Density (Mg/m ³) (Total Pressure (GPa)- and Temperature (K)-dependent)	301
EOS_Gs_D		ES4_SHEARM	Shear Modulus (Gpa) (Density (Mg/m ³)-dependent)	431
EOS_Kc_DT	«	ES4_OPACC3	Electron Conductive Opacity (Conductivity Model) (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)-dependent)	605
EOS_Kec_DT	«	ES4_ECONDE	Electrical Conductivity (1/s) (Density (Mg/m ³)- and Temperature (eV)-dependent)	602

⁹ This is the proposed final list as of this writing (1/5/2011 12:37 PM).

EOSPAC 6 Constant		EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_Keo_DT	«	ES4_OPACC2	Electron Conductive Opacity (Opacity Model) (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)-dependent)	503
EOS_Kp_DT	«	ES4_OPACP	Planck Mean Opacity (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)-dependent)	505
EOS_Kr_DT	«	ES4_OPACR	Rosseland Mean Opacity (cm ² /g) (Density (Mg/m ³)- and Temperature (eV)-dependent)	502
EOS_Ktc_DT	«	ES4_TCONDE	Thermal Conductivity (1/cm/s) (Density (Mg/m ³)- and Temperature (eV)-dependent)	603
EOS_NullTable		ES4_NULLPTR	null table	n/a
EOS_Pc_D	«	ES4_PRCLD	Pressure Cold Curve (GPa) (Density (Mg/m ³)-dependent)	306
EOS_Pe_DT	«	ES4_PRELC	Electron Pressure (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	304
EOS_Pe_DUe	«	ES4_PNELC	Electron Pressure (GPa) (Density (Mg/m ³)- and Electron Specific-Internal- Energy (MJ/kg)-dependent)	304
EOS_Pf_D		ES4_PFREEZ	Freeze Pressure (GPa) (Density (Mg/m ³)-dependent)	412
EOS_Pic_DT	«	ES4_PRION	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	303

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_Pic_DUic «	ES4_PNION	Ion Pressure plus Cold Curve Pressure (GPa) (Density (Mg/m ³)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg)-dependent)	303
EOS_Pm_D	ES4_PMELT	Melt Pressure (GPa) (Density (Mg/m ³)-dependent)	411
EOS_Pt_DT «	ES4_PRTOT	Total Pressure (GPa) (Density (Mg/m ³)- and Temperature (K)-dependent)	301
EOS_Pt_DUt «	ES4_PNTOT	Total Pressure (GPa) (Density (Mg/m ³)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_T_DPe «	ES4_TPELC	Temperature (K) (Density (Mg/m ³)- and Electron Pressure (GPa)-dependent)	304
EOS_T_DPic «	ES4_TPION	Temperature (K) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303
EOS_T_DPt «	ES4_TPTOT	Temperature (K) (Density (Mg/m ³)- and Total Pressure (GPa)-dependent)	301
EOS_T_DUe «	ES4_TNELC	Temperature (K) (Density (Mg/m ³)- and Electron Specific-Internal-Energy (MJ/kg)-dependent)	304
EOS_T_DUic «	ES4_TNION	Temperature (K) (Density (Mg/m ³)- and Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-	303

EOSPAC 6 Constant	EOSPAC 5 Constant	Description	SESAME Table(s)
		Energy (MJ/kg)-dependent)	
EOS_T_DUt «	ES4_TNTOT	Temperature (K) (Density (Mg/m ³)- and Total Specific-Internal-Energy (MJ/kg)-dependent)	301
EOS_Tf_D	ES4_TFREEZ	Freeze Temperature (eV) (Density (Mg/m ³)-dependent)	412
EOS_Tm_D	ES4_TMELT	Melt Temperature (K) (Density (Mg/m ³)-dependent)	411
EOS_Uc_D «	ES4_ENCLD	Specific-Internal-Energy Cold Curve (MJ/kg) (Density (Mg/m ³)-dependent)	306
EOS_Ue_DPe «	ES4_EPELC	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Electron Pressure (GPa)-dependent)	304
EOS_Ue_DT «	ES4_ENELC	Electron Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	304
EOS_Uf_D	ES4_EFREEZ	Freeze Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)-dependent)	412
EOS_Uic_DPic «	ES4_EPION	Ion Specific-Internal-Energy plus Cold Curve Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Ion Pressure plus Cold Curve Pressure (GPa)-dependent)	303

EOSPAC 6 Constant		EOSPAC 5 Constant	Description	SESAME Table(s)
EOS_Uic_DT	«	ES4_ENION	Ion Specific-Internal-Energy plus Cold Curve Specific- Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	303
EOS_Um_D		ES4_EMELT	Melt Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)-dependent)	411
EOS_Ut_DPt	«	ES4_EPTOT	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Total Pressure (GPa)-dependent)	301
EOS_Ut_DT	«	ES4_ENTOT	Total Specific-Internal-Energy (MJ/kg) (Density (Mg/m ³)- and Temperature (K)-dependent)	301
EOS_Zfc_DT	«	ES4_ZFREE3	Mean Ion Charge (Conductivity Model) (free electrons per atom) (Density (Mg/m ³)- and Temperature (eV)-dependent)	601
EOS_Zfo_DT	«	ES4_ZFREE2	Mean Ion Charge (Opacity Model) (free electrons per atom) (Density (Mg/m ³)- and Temperature (eV)-dependent)	504

APPENDIX D. SETUP PHASE OPTION FLAG DEFINITIONS

Below is a list of defined constants¹⁰ corresponding to the user specified setup phase options available within EOSPAC. This list has been alphabetized according to the defined constant names, which are cross-referenced to the applicable EOSPAC 5 defined constants. Unlike EOSPAC 5, these EOSPAC option flags are to be applied to a given table handle using one of two public routines: `eos_ResetOption` and `eos_SetOption` (see sections 7.1.5 and 7.1.8 respectively). For each table handle, the `eos_SetOption` routine may be used to enable or disable an optional feature. Alternatively, the `eos_ResetOption` routine may be used to reassert the default option settings if, in fact, such default values are defined in the table below. *Take note that some of the options require an associated value passed into the `eos_SetOption` routine parameter named `tableOptionVal`.*

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

EOSPAC 6 Constant	Default Option State (<code>tableOptionVal</code>)	Description
EOS_ADJUST_VAP_PRES	Disabled (N/A)	This provides a mechanism for the host code to pass into EOSPAC 6 adjusted pressure values (corresponding to SAGE's ¹¹ <code>matdef(2,mat)</code> input variable) for the vapor dome to ensure ambient conditions are reasonable for a specified material. This option is only valid when also using the option named EOS_PT_SMOOTHING. It is important to note that the units of the <code>tableOptionVal</code> must be compatible with Sesame pressure data (GPa). See section 8.1 for more details.

¹⁰ This is by no means a complete list as of this writing (1/5/2011 12:37 PM).

¹¹ SAGE is a one-, two-, and three-dimensional, multi-material Eulerian hydrodynamics code (LA-UR-04-2959).

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_APPEND_DATA	Disabled (N/A)	Append the loaded data table and descriptive information to an ASCII file named "TablesLoaded.dat" within the current working directory. The corresponding EOSPAC 5 setup option used to enable this feature is <code>lprnt=TRUE</code> passed to ES1TABS.
EOS_CALC_FREE_ENERGY	Disabled (N/A)	Instead of using the corresponding Sesame data, the Helmholtz Free Energy data is calculated using the equations (2.1), (2.2) and (2.3). If no internal energy data exists for $T=0$, then the free energy data will not be calculated.
EOS_CHECK_ARGS ¹²	Disabled (N/A)	Allow extensive argument checking.
EOS_DUMP_DATA	Disabled (N/A)	Write the loaded data table and descriptive information to an ASCII file named "TablesLoaded.dat" within the current working directory. The corresponding EOSPAC 5 setup option used to enable this feature is <code>lprnt=TRUE</code> passed to ES1TABS.
EOS_INSERT_DATA ¹³	Disabled (0)	Insert grid points between each original grid point with respect to all independent variables (i.e., increase grid resolution). The value of the <code>eos_SetOption</code> parameter, <code>tableOptionVal</code> , is to contain the user-defined number of data points to insert between existing data points. The corresponding EOSPAC 5 setup option used to enable this feature is <code>iopt = 10000*N</code> , given $(0 < N <= 9)$ passed to ES1TABS.

¹² As of this writing (1/5/2011 12:37 PM), this option has not yet been implemented.

¹³ The `eos_SetOption` parameter, `tableOptionVal` (see section 7.1.7), must be defined to an appropriate number for this option.

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_MONOTONIC_IN_X	Disabled (N/A)	Enable forced monotonicity with respect to x of F(x,y). The corresponding EOSPAC 5 setup option used to enable this feature is iopt=100 passed to ES1TABS.
EOS_MONOTONIC_IN_Y	Disabled (N/A)	Enable forced monotonicity with respect to y of F(x,y). The corresponding EOSPAC 5 setup option used to enable this feature is iopt=300 passed to ES1TABS.
EOS_PT_SMOOTHING	Disabled (N/A)	This performs all the necessary data smoothing taken from SAGE. ¹⁴ See the related setup option named EOS_ADJUST_VAP PRES and the related interpolation option named EOS_USE_CUSTOM_INTERP. See section 8.1 for more details.
EOS_SMOOTH	Disabled (N/A)	Enable data table smoothing that imposes a linear floor on temperature dependence, forces linear temperature dependence for low temperature, and forces linear density dependence for low and high density. The corresponding EOSPAC 5 setup option used to enable this feature is iopt=10 passed to ES1TABS.
EOS_SPLIT_COWAN	Disabled (N/A)	Allows splitting for ion data table not found in the database using the cold curve plus Cowan-nuclear model for ions.
EOS_SPLIT_FORCED	Disabled (N/A)	Forces specified splitting option for data table.
EOS_SPLIT_IDEAL_GAS	Disabled (N/A)	Allows splitting for ion data table not found in the database using the cold curve plus ideal gas model for ions.

¹⁴ SAGE is a one-, two-, and three-dimensional, multi-material Eulerian hydrodynamics code (LA-UR-04-2959).

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_SPLIT_NUM_PROP	Disabled (N/A)	Allows splitting for ion data table not found in the database using the cold curve plus number-proportional model for ions.

APPENDIX E. DATA INFORMATION PARAMETERS

Information about a table can be requested via the eos_GetTableInfo routine using the parameters defined in this appendix. The eos_GetTableInfo routine is designed to be general in functionality, so these parameters are grouped according to their prerequisites.

Table E-1 list parameters that require the comment tables (i.e., EOS_Comment) for a material to be loaded and associated with a table handle.

Table E-1 Information parameter(s) related to SESAME's 100-series tables

Parameter	Description
EOS_Cmnt_Len	Retrieve the length in characters of the comments available for the specified data table

Table E-2 list parameters that require the general material data table (i.e., EOS_Info) to be loaded and associated with a table handle.

Table E-2 Information parameter(s) related to SESAME's 201 tables

Parameter	Description
EOS_Exchange_Coeff	Retrieve the exchange coefficient
EOS_Mean_Atomic_Mass	Retrieve the mean atomic mass
EOS_Mean_Atomic_Num	Retrieve the mean atomic number
EOS_Modulus	Retrieve the solid bulk modulus
EOS_Normal_Density	Retrieve the normal density

Table E-3 list parameters that require data to be loaded and associated with a table handle; however, they don't apply to SESAME's 100-series and 201 tables.

Table E-3 Information parameter(s) generally related to SESAME's tables except for SESAME's 100-series and 201 tables

Parameter	Description
EOS_F_Convert_Factor	Retrieve the conversion factor corresponding to the dependent variable, $F(x,y)$
EOS_Log_Val	Retrieve the InfoVal that is non-zero if the data table is in a \log_{10} format
EOS_X_Convert_Factor	Retrieve the conversion factor corresponding to the primary independent variable, x
EOS_Y_Convert_Factor	Retrieve the conversion factor corresponding to the secondary independent variable, y

Table E-4 list parameters that require data to be loaded and associated with a table handle. In other words, all table handles that are associated with data may be queried for the information indicated by these parameters.

Table E-4 Information parameter(s) generally related to SESAME's tables

Parameter	Description
EOS_Material_ID	Retrieve the SESAME material identification number
EOS_Table_Type	Retrieve the type of data table. Corresponds to the parameters in APPENDIX B and APPENDIX C

Table E-5 list parameters that require data to be loaded and associated with a table handle; however, they are only valid for non-inverted data tables specifically related to SESAME's 301 and 401 tables.

Table E-5 Information parameter(s) associated with non-inverted data tables

Parameter	Description
EOS_R_Array	Retrieve the density array Note that InfoVals must be allocated to hold NR EOS_REAL values, so querying for the EOS_NR value is first necessary.
EOS_T_Array	Retrieve the temperature array Note that InfoVals must be allocated to hold NT EOS_REAL values, so querying for the EOS_NT value is first necessary.
EOS_F_Array	Retrieve the F array This two-dimensional array will be assigned to the one-dimensional array, InfoVals, in a column-major order. Note that InfoVals must be allocated to hold NR*NT EOS_REAL values, so querying for the EOS_NR and EOS_NT values is first necessary.
EOS_NR	Retrieve the number of densities
EOS_NT	Retrieve the number of temperatures
EOS_Rmin	Retrieve the minimum density
EOS_Rmax	Retrieve the maximum density
EOS_Tmin	Retrieve the minimum temperature
EOS_Tmax	Retrieve the maximum temperature
EOS_Fmin	Retrieve the minimum F value
EOS_Fmax	Retrieve the maximum F value
EOS_NUM_PHASES	Retrieve the number of material phases that are tabulated. This is only valid in conjunction with the EOS_M_DT data type.

APPENDIX F. INTERPOLATION PHASE OPTION FLAG DEFINITIONS

Below is a list of defined constants¹⁵ corresponding to the user specified interpolation options available within EOSPAC. This list has been alphabetized according to the defined constant names, which are cross-referenced to the applicable EOSPAC 5 defined constants. Unlike EOSPAC 5, these EOSPAC option flags are to be applied to a given table handle using one of two public routines: `eos_ResetOption` and `eos_SetOption` (see sections 7.1.5 and 7.1.8 respectively). For each table handle, the `eos_SetOption` routine may be used to enable or disable an optional feature. Alternatively, the `eos_ResetOption` routine may be used to reassert the default option settings if, in fact, such default values are defined in the table below. *Take note that some of the options require an associated value passed into the `eos_SetOption` routine parameter named `tableOptionVal`.*

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

EOSPAC 6 Constant	Default Option State (<code>tableOptionVal</code>)	Description
<code>EOS_CHECK_ARGS</code> ¹⁶	Disabled (N/A)	Allow extensive argument checking.
<code>EOS_F_CONVERT</code> ¹⁷	Disabled (1.0)	Set the conversion factor used on the <code>fVals</code> dependent variable value(s). The value of the <code>eos_SetOption</code> parameter, <code>tableOptionVal</code> , is to contain the conversion factor value.
<code>EOS_LINEAR</code>	Disabled (N/A)	Bilinear interpolation. The corresponding EOSPAC 5 interpolation option used to enable this feature is <code>idrvs=ES4_BILINE</code> passed to <code>ES1VALS</code> .

¹⁵ This is not necessarily a complete list as of this writing (1/5/2011 12:37 PM).

¹⁶ As of this writing (1/5/2011 12:37 PM), this option has not yet been implemented.

¹⁷ The `eos_SetOption` parameter, `tableOptionVal` (see section 7.1.7), must be defined to an appropriate number for this option.

EOSPAC 6 Constant	Default Option State (tableOptionVal)	Description
EOS_RATIONAL	Enabled (N/A)	Birational interpolation. The corresponding EOSPAC 5 interpolation option used to enable this feature is idrvs=ES4_BIRATF passed to ES1VALS.
EOS_USE_CUSTOM_INTERP	Disabled (N/A)	Use a custom inverse-interpolation algorithm that requires the setup option, EOS_PT_SMOOTHING, to be enabled for the specified table handle. This option is only valid for table types EOS_Ut_PtT and EOS_V_PtT. See section 8.1 for more details. Note that the partial derivatives, dFx and dFy, are <i>not</i> calculated when this option is set.
EOS_X_CONVERT ¹⁷	Disabled (1.0)	Set the conversion factor used on the xVals independent variable value(s). The value of the eos_SetOption parameter, tableOptionVal, is to contain the conversion factor value.
EOS_Y_CONVERT ¹⁷	Disabled (1.0)	Set the conversion factor used on the yVals independent variable value(s). The value of eos_SetOption parameter, tableOptionVal, is to contain the conversion factor value.

APPENDIX G. ERROR CODE DEFINITIONS

Below is a list of defined constants¹⁸ corresponding to all of the possible error codes returned by EOSPAC. This list has been alphabetized according to the defined constant names, which are cross-referenced to the applicable EOSPAC 5 defined constants.

It is important to note that the actual values of these constants may change without notice; therefore, use the constant names – do not hardwire the values into the host code.

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_BAD_DATA_TYPE (ES5_BADTABLETYPE)	Data table type is not recognized
EOS_BAD_DERIVATIVE_FLAG (ES5_BADDERIVTYPE)	Derivative is not recognized
EOS_BAD_INTERPOLATION_FLAG (ES5_BADINTRPTYPE)	Interpolation is not recognized
EOS_BAD_MATERIAL_ID (ES5_MATIDZERO)	Material ID is zero
EOS_CANT_INVERT_DATA	Can't invert with respect to the required independent variable
EOS_CANT_MAKE_MONOTONIC	Can't make data monotonic in X
EOS_CONVERGENCE_FAILED (ES5_CONVERGEFAILED)	Iterative algorithm did not converge during inverse interpolation
EOS_DATA_TYPE_NOT_FOUND (ES5_TYPEROTFOUND)	Data table type is not in library
EOS_DATA_TYPE_NO_MATCH	Data types do not match as required for mixing
EOS_FAILED	Operation failed
EOS_INTEGRATION_FAILED	Numerical integration failed or not possible
EOS_INTERP_EXTRAPOLATED	Interpolation caused extrapolation beyond data table boundaries
EOS_INTERP_EXTRAP_PBAL	Pressure balance function extrapolated beyond data table boundaries

¹⁸ This is not necessarily a complete list as of this writing (1/5/2011 12:37 PM).

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_INTERP_EXTRAP_TBAL	Temperature balance function extrapolated beyond data table boundaries
EOS_INVALID_CONC_SUM	The sum of the supplied material concentrations does not equal 1.0
EOS_INVALID_DATA_TYPE	Operation is not defined on this data type
EOS_INVALID_INFO_FLAG	The info flag passed into eos_GetTableInfo() is invalid
EOS_INVALID_NXYPAIRS	Invalid nXYPairs value
EOS_INVALID_OPTION_FLAG	The option flag passed into eos_SetOption() is invalid
EOS_INVALID_SPLIT_FLAG	The data splitting option is invalid
EOS_INVALID_SUBTABLE_INDEX	Subtable index out of the range
EOS_INVALID_TABLE_HANDLE	Invalid table handle
EOS_MATERIAL_NOT_FOUND (ES5_MATNOTFOUND)	Material ID is not in library
EOS_MEM_ALLOCATION_FAILED (ES5_EXPANDFAILED)	EOS table area cannot be expanded
EOS_MIN_ERROR_CODE_VALUE	Minimum error code value
EOS_NOT_ALLOCATED	Memory not allocated for data
EOS_NOT_INITIALIZED (ES5_NOTINIT)	EOS table area is not initialized
EOS_NO_COMMENTS	No comments available for this data table
EOS_NO_DATA_TABLE (ES5_NOTABLE)	Data table is not in EOS table area
EOS_NO_SESAME_FILES (ES5_NOFILESFOUND)	No data library files exist
EOS_OK (ES5_OK)	No errors detected
EOS_OPEN_OUTPUT_FILE_FAILED	Could not open TablesLoaded.dat or related data file

EOSPAC 6 Constant (EOSPAC 5 Constant)	Description
EOS_OPEN_SESAME_FILE_FAILED (ES5_OPENFAILED)	Could not open data file
EOS_READ_DATA_FAILED (ES5_LDTABLEFAILED)	Could not load data table
EOS_READ_FILE_VERSION_FAILED (ES5_GETVERSNFAILED)	Could not load version from data file
EOS_READ_MASTER_DIR_FAILED (ES5_LDMASTERFAILED)	Could not load master directory
EOS_READ_MATERIAL_DIR_FAILED (ES5_LDMATDIRFAILED)	Could not load material directory
EOS_READ_TOTAL_MATERIALS_FAILED (ES5_GETNMATSFALLED)	Could not read number of materials
EOS_SPLIT_FAILED	The data splitting algorithm failed
EOS_UNDEFINED	The result is undefined
EOS_WARNING	Operation has generated a warning and an associated custom message
EOS_xHi_yHi	Both the x and y arguments were high
EOS_xHi_yLo	The x argument was high, the y argument was low ¹⁹
EOS_xHi_yOk	The x argument was high, the y argument was OK ²⁰
EOS_xLo_yHi	The x argument was low, the y argument was OK ²⁰
EOS_xLo_yLo	Both the x and y arguments were low ^{19,20}
EOS_xLo_yOk	The x argument was low, the y argument was OK ²⁰
EOS_xOk_yHi	The x argument is OK and the y argument is high
EOS_xOk_yLo	The x argument is OK and the y argument is low ¹⁹

¹⁹ If the y argument corresponds to a temperature value, then a zero temperature was used for interpolation rather than the value supplied by the host code.

²⁰ If the x argument corresponds to a density value, then a zero density was used for interpolation rather than the value supplied by the host code.

January 5, 2011

EOSPAC User's Manual: Version 6.2