



Aspen Plus

User Models

Version Number: V8.2

May 2013

Copyright (c) 1981-2013 by Aspen Technology, Inc. All rights reserved.

Aspen Plus, aspenONE, the aspen leaf logo and Plantelligence and Enterprise Optimization are trademarks or registered trademarks of Aspen Technology, Inc., Burlington, MA.

All other brand and product names are trademarks or registered trademarks of their respective companies.

This software includes NIST Standard Reference Database 103b: NIST Thermodata Engine Version 7.1

This document is intended as a guide to using AspenTech's software. This documentation contains AspenTech proprietary and confidential information and may not be disclosed, used, or copied without the prior consent of AspenTech or as set forth in the applicable license agreement. Users are solely responsible for the proper use of the software and the application of the results obtained.

Although AspenTech has tested the software and reviewed the documentation, the sole warranty for the software may be found in the applicable license agreement between AspenTech and the user. ASPENTECH MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESSED OR IMPLIED, WITH RESPECT TO THIS DOCUMENTATION, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

Aspen Technology, Inc.

200 Wheeler Road

Burlington, MA 01803-5501

USA

Phone: (1) (781) 221-6400

Toll Free: (1) (888) 996-7100

URL: <http://www.aspentech.com>

Contents

Who Should Read this Guide	1
Introducing Aspen Plus	3
Related Documentation	4
Technical Support	5
1 Writing and Using User Models	7
Fortran User Models	8
Moving to the Intel Fortran Compiler	9
Configuring Aspen Plus for Your Fortran Compiler	9
Compiler-Compatible Write Statements	10
Writing Fortran User Models	11
Dynamic Linking Overview	12
Compiling Fortran User Models	13
Supplying Fortran User Models to Aspen Plus	13
Creating Fortran Shared Libraries Using Asplink	14
Writing DLOPT Files	14
Specifying DLOPT Files for Aspen Plus Runs	15
Modifying Asplink	16
Using the IMSL Library with Aspen Plus	17
No Recursive Calls	20
2 Calling the Flash Utility	21
Flash Utility FLSH_FLASH	21
NBOPST	24
Flash Types	24
RETN	24
IRETN	25
Flash Results Stored in COMMON	25
3 Calling Physical Property Monitors	27
Calling Sequences for Thermodynamic Property Monitors	28
Calling Sequences for Transport Property Monitors	30
Calling Sequences for Nonconventional Property Monitors	31
Calling Sequences for Thermodynamic and Transport Property Monitors with Derivatives	31
Argument Descriptions for Physical Property Monitors	31
IDX	34
IDXNC	34
Y, X, X1, X2, Z, CS	34
CAT	35
NBOPST	35

KDIAG	35
Calculation Codes.....	35
Phases.....	36
CALPRP Results.....	36
Derivatives.....	36
Calling Sequences for PROP-SET Property Monitors	37
Argument Descriptions for PROP-SET Property Monitors	38
PROPS	39
PHASES	39
KWDBS.....	39
XPCLV.....	40
KULAB	40
CALUPP Results.....	40
Example of Calling CALUPP Multiple Times to Retrieve Multiple Properties.....	41
4 Calling Utility Subroutines.....	45
Packing Utilities	47
Aspen Plus Error Handler.....	48
Report Header Utility	49
ISECT	50
Terminal File Writer Utility.....	51
Utilities to Determine Component Index.....	52
Component Index Example.....	53
CAS Number Utility.....	53
Component Attribute Information Utilities	54
Component Attribute Calculation Utilities.....	57
Polymer Property Utilities	59
Polymer Type Utilities	63
Polymer Component Fraction Utilities.....	65
General Stream Handling Utilities.....	66
Plex Offset Utility	69
Ambient Pressure Utility.....	70
Break Utility	71
License Name Utility	71
The Fortran WRITE Statement	71
5 User Unit Operation Models.....	75
User and User2 Fortran Models	75
Stream Structure and Calculation Sequence	78
NBOPST	78
Size.....	79
Integer and Real Parameters	79
Local Work Arrays	80
Simulation Control Guidelines	80
History File.....	81
Terminal File.....	82
Report File	82
Control Panel	82
Incorporating Excel Worksheets into User2.....	82
Extending the User2 Concept.....	83
Excel File Name	83
Fortran Routine.....	83

The Excel Template	83
Tables.....	84
The Helper Functions	85
The Hook Functions	88
The Sample Workbook	89
Creating or Converting Your Own Excel Models	90
Converting an Existing Excel Model.....	91
Customizing the Fortran Code.....	92
Accessing User2 Parameters	96
Accessing parameters by position.....	96
Accessing parameters by name	97
Other Helper Functions	99
User3 Fortran Models.....	100
Stream Structure and Calculation Sequence	103
NBOPST	104
USER3 Data Classifications	104
Size.....	105
Variable Types and Mapping Concepts	106
Scaling and Units Conversion in USER3	108
K Codes	108
Sparsity Example	110
Creating a USER3 Model.....	111
Additional User3 Subroutines.....	113
Physical Property Call Primitives.....	120
Low-Level Physical Property Subroutines.....	123
Other Useful USER3 Utilities	125
Component Object Models (COM)	126

6 User Physical Property Models127

User Models for Conventional Properties.....	128
Principal User Model Subroutines for Conventional Properties.....	132
IDX	138
Partial Component Index Vectors	138
X, Y, Z	138
Real and Integer Work Areas	138
KOP.....	139
KDIAG	139
Calculation Codes.....	139
Range of Applicability	140
Units of Measurement.....	140
Global Physical Property Constants	140
User K-Value	140
Electrolyte Calculations	141
Model-Specific Parameters for Conventional Properties	141
Universal Constant Names and Definitions.....	141
Naming Model-Specific Parameters	142
Multiple Data Sets	142
Parameter Retrieval.....	143
User Models for Nonconventional Properties.....	145
Using Component Attributes	145
Principal User Model Subroutines for Nonconventional Properties.....	146
IDXNC	147

Real and Integer Work Areas	147
KOP	147
KDIAG	148
Range of Applicability	148
Model-Specific Parameters for Nonconventional Properties	148
Naming Model-Specific Parameters	149
Accessing Component Attributes	149
7 User Properties for Property Sets.....	151
Subroutine to Define a User Property	151
IDX	153
NBOPST, KDIAG, and KPDIAG.....	153
Phase Codes	153
Passing Phase Fraction and Composition Information	153
Component Order Independence	154
8 User Stream Report.....	155
Stream Report Subroutine.....	155
Stream Classes	156
PRPVAL.....	156
NRPT	157
Component IDs.....	157
9 User Blending Subroutines for Petroleum Properties	159
Petroleum Blending Subroutine	159
IDX	160
10 User Subroutines for Sizing and Costing.....	161
Sizing Model Subroutine.....	161
Integer and Real Parameters	162
ICSIZ	162
IRSLT	162
Costing Model Subroutine.....	163
11 User Kinetics Subroutines	165
Kinetics Subroutine for USER Reaction Type	166
Integer and Real Parameters	169
NBOPST	169
Local Work Arrays	169
Calling Model Type	170
STOIC.....	170
Reaction Rates.....	170
Component Attributes and Substream PSD	171
User Variables	171
Component Fluxes in RPlug	172
COMMON RPLG_RPLUGI.....	172
COMMON RPLG_RPLUGR	173
COMMON RBTC_RBATI.....	173
COMMON RBTC_RBATR.....	173
COMMON RCST_RCSTRI.....	174

COMMON RXN_RCSTRR	174
COMMON PRSR_PRESRI	174
COMMON PRSR_PRESRR	175
COMMON RXN_DISTI	175
COMMON RXN_DISTR	175
COMMON RXN_RPROPS	176
User Kinetics Subroutine for REAC-DIST Reaction Type	176
NBOPST	178
STOIC	178
12 User Pressure Drop and Holdup Subroutines for RPlug	179
RPlug Pressure Drop Subroutine	180
RPlug Holdup Subroutine	182
NBOPST	183
Integer and Real Parameters	183
Pressure	183
Local Work Arrays	184
User Variables (Pressure Drop Subroutine only)	184
COMMON RPLG_RPLUGI	184
COMMON RPLG_RPLUGR	185
13 User Heat Transfer Subroutine for RPlug	187
RPlug Heat Transfer Subroutine	188
NBOPST	190
Integer and Real Parameters	190
Local Work Arrays	190
User Variables	190
COMMON RPLG_RPLUGI	190
COMMON RPLG_RPLUGR	191
Heat Flux Terms	192
14 User Heat Transfer Subroutine for RBatch	193
RBatch Heat Transfer Subroutine	194
Stream Structure	195
NBOPST	195
Integer and Real Parameters	195
Local Work Arrays	195
User Variables	195
15 User Subroutines for RYield	197
RYield User Subroutines	198
NBOPST	200
Integer and Real Parameters	200
Local Work Arrays	200
16 User KLL Subroutines	201
User KLL Subroutine	202
IDX	203
X1, X2	203
NBOPST, KDIAG	203

Component Sequence Number	203
Integer and Real Parameters	203
17 User Subroutines for Pipes and HeatX.....	205
User Pressure Drop Subroutine	206
User Liquid Holdup Subroutine	208
User Diameter Subroutine	209
Integer and Real Parameters	211
NBOPST	211
Local Work Arrays	211
KFLASH.....	211
18 User Heat Transfer Subroutine for HeatX	213
HeatX Heat Transfer Subroutine.....	213
NBOPST	215
Integer and Real Parameters	215
Local Work Arrays	215
Equipment Specification Arrays	215
19 User LMTD Correction Factor Subroutine for HeatX	219
HeatX LMTD Correction Factor Subroutine	220
NBOPST	221
Integer and Real Parameters	221
Local Work Arrays	221
Equipment Specification Arrays	222
20 User Subroutines for Rate-Based Distillation.....	223
Rate-Based Binary Mass Transfer Coefficient Subroutine	224
Integer and Real Parameters	226
Rate-Based Heat Transfer Coefficient Subroutine	229
Integer and Real Parameters	230
Rate-Based Interfacial Area Subroutine.....	232
Integer and Real Parameters	234
Rate-Based Holdup Subroutine.....	236
Integer and Real Parameters	237
Packing Parameters	240
Packing Type Specification	241
Packing Vendor Specification	241
Packing Material Specification	242
Packing Size Specification	242
21 User Tray and Packing Subroutines	243
User Tray Sizing/Rating Subroutine	244
RTPAR	246
RTRSLT	246
User Packing Sizing/Rating Subroutine.....	247
Integer and Real Parameters	248
22 User Performance Curves Subroutine for Compr/MCompr.....	249
Performance Curve Subroutine	250

Integer and Real Parameters	251
Local Work Arrays	251
23 User Solubility Subroutine for Crystallizer.....	253
Crystallizer Solubility Subroutine.....	253
IDX	255
Integer and Real Parameters	255
Local Work Arrays	255
24 User Performance Curves Subroutine for Pump.....	257
Pump Performance Curve Subroutine.....	257
Integer and Real Parameters	258
Local Work Arrays	259
25 User Subroutines for Petroleum Property Methods.....	261
26 COM Unit Operation Interfaces.....	269
Components and Interfaces	270
Unit Interfaces.....	273
ICapeUnit Interface Methods	273
ICapeUtilities Interface Method	276
Port Interfaces.....	279
ICapeUnitPort Interface Methods.....	279
Parameter Interfaces	282
ICapeParameter	282
ICapeParameterSpec	285
ICapeRealParameterSpec Interface Methods.....	286
ICapeIntegerParameterSpec Interface Methods.....	288
ICapeOptionParameterSpec Interface Methods.....	290
Collection Interfaces	292
ICapeCollection Interface Methods.....	292
ICapeIdentification Interface Methods.....	293
ComponentDescription	294
ComponentName	294
Aspen Plus Interfaces	294
IATCapeXDiagnostic Interface Methods	295
IATCapeXRealParameterSpec.....	296
Installation of COM Unit Operations	300
Distributing COM Models to Users.....	301
Adding Compiled COM Models to the Aspen Plus Model Palette.....	301
Version Compatibility for Visual Basic COM Models	302
Uninstalling COM Models	302
27 CAPE-OPEN COM Thermodynamic Interfaces.....	303
Material Templates	306
Material Objects	307
ICapeThermoMaterialObject Interface Methods	307
Physical Property System	315
ICapeThermoSystem Interface Methods.....	316
Property Package	317
Importing and Exporting	317

ICapeThermoPropertyPackage Interface Methods	318
Registration of CAPE-OPEN Components	323
28 COM Interface for Updating Oil Characterizations and Petroleum Properties	325
IAssayUpdate Interface Methods	326
Modifying Petroleum Properties During a Simulation	326
Recalculate Characterization Parameters	329
Additional IAssayUpdate Interface Methods	331
A Common Blocks and Accessing Component Data.....	337
Aspen Plus Common Blocks	338
COMMON DMS_ERROUT	338
COMMON DMS_FLSCOM	338
COMMON DMS_NCOMP	339
COMMON DMS_PLEX	339
COMMON PPUTL_PPGLOB	340
COMMON DMS_RGLOB.....	341
COMMON DMS_RPTGLB	341
COMMON DMS_STWKWK	342
COMMON SHS_STWORK	342
COMMON PPEXEC_USER	344
Accessing Component Data Using the Plex	345
FRMULA	345
IDSCC	346
IDSNCC	346
IDXNCC	347
Paramname	347
Using IPOFF3	348
Accessing Component Data using PPUTL_GETPARAM	349
B User Subroutine Templates and Examples.....	351
C Stream Structure.....	353
Substream MIXED	354
Substream CISOLID	354
Substream NC	355
Determining Particle Size Distribution Length.....	356

Who Should Read this Guide

This manual is intended for the Aspen Plus user who wants to create custom Fortran subroutines and CAPE-OPEN models to extend the modeling capabilities of Aspen Plus. Programming experience with Fortran, C++, or Visual Basic is recommended.

Introducing Aspen Plus

This volume of the Aspen Plus Reference Manuals, *User Models*, describes how to write an Aspen Plus user model when the built-in models provided by Aspen Plus do not meet your needs. For example, you can write a user model for a complete unit operation model, a complete physical property model, a sizing and costing model, special stream properties or stream reports, or to add calculations to built-in Aspen Plus unit operation models.

An Aspen Plus user model consists of one or more Fortran subroutines. Experienced Aspen Plus users with a knowledge of Fortran programming find user models a very powerful tool for customizing process models.

A CAPE-OPEN user model consists of a collection of routines implementing one or more interfaces defined by the CAPE-OPEN standard. See chapters 26-28 for more information about these subroutines.

Related Documentation

Title	Content
Aspen Plus Getting Started Building and Running a Process Model	Tutorials covering basic use of Aspen Plus. A prerequisite for the other Getting Started guides
Aspen Plus Getting Started Modeling Processes with Solids	Tutorials covering the Aspen plus features designed to handle solids
Aspen Plus Getting Started Modeling Processes with Electrolytes	Tutorials covering the Aspen plus features designed to handle electrolytes
Aspen Plus Getting Started Using Equation-Oriented Modeling	Tutorials covering the use of equation-oriented models in Aspen Plus
Aspen Plus Getting Started Customizing Unit Operation Models	Tutorials covering the development of custom unit operation models in Aspen Plus
Aspen Plus System Management Reference Manual	Information about customizing files provided with Aspen Plus
Aspen Plus Summary File Toolkit Reference Manual	Information about the Summary File Toolkit, a library designed to read Aspen Plus summary files.
Aspen Plus Input Language Guide Reference Manual	Syntax and keyword meanings for the Aspen Plus input language, and accessible variables.
OOMF Script Language Reference Manual	Syntax and command meanings for the OOMF Script language
APrSystem Physical Property Methods and Physical Property Models Reference Manuals	Information about property methods and property models
Aspen Plus Application Examples	A suite of examples illustrating capabilities of Aspen Plus
Aspen Engineering Suite Installation Manual	Instructions for installing Aspen Plus and other Aspen Engineering Suite products

Technical Support

AspenTech customers with a valid license and software maintenance agreement can register to access the online AspenTech Support Center at:

<http://support.aspentech.com>

This Web support site allows you to:

- Access current product documentation
- Search for tech tips, solutions and frequently asked questions (FAQs)
- Search for and download application examples
- Search for and download service packs and product updates
- Submit and track technical issues
- Send suggestions
- Report product defects
- Review lists of known deficiencies and defects

Registered users can also subscribe to our Technical Support e-Bulletins. These e-Bulletins are used to alert users to important technical support information such as:

- Technical advisories
- Product updates and releases

Customer support is also available by phone, fax, and email. The most up-to-date contact information is available at the AspenTech Support Center at

<http://support.aspentech.com>.

1 Writing and Using User Models

Aspen Plus provides several methods for creating customized models:

Method	Reference
Fortran	See Fortran User Models in this chapter.
Excel	See Chapter 5 for Excel unit operation models. See the chapter "Calculator Blocks and In-line Fortran" in the Aspen Plus User Guide for Excel Calculator blocks.
COM Models based on the CAPE-OPEN standard	See Chapter 26.
Aspen Custom Modeler	See ACM documentation.

Fortran User Models

This section describes how to write and compile Fortran user models, and how to specify the location of the Fortran user models to use during Aspen Plus runs.

An Aspen Plus Fortran user model consists of one or more subroutines that you write yourself to extend the capabilities of Aspen Plus. You can write six kinds of Fortran user models for use in Aspen Plus:

- User unit operation models.
- User physical property models for calculating the various major, subordinate, and intermediate physical properties.
- User models for sizing and costing.
- User models for special stream properties.
- User stream reports.
- User models for performing various types of calculations within Aspen Plus unit operation models.

Examples of calculations that Fortran user models perform within Aspen Plus unit operation models include:

- Reaction rates.
- Heat transfer rates/coefficients.
- Pressure drop.
- Liquid-liquid distribution coefficients.

Fortran user models can call:

- Aspen Plus utility routines to perform flash and physical property calculations.
- The Aspen Plus error handler to report errors in calculations.

Chapters 2 through 4 in this manual describe the Aspen Plus subroutines that Fortran user models can call. All chapters in this manual describe the proper argument list you need to declare in your subroutines to interface your user model to Aspen Plus. The Argument List Descriptions describe the input and/or output variables to the subroutines.

Throughout this chapter, the term “shared library” will be used in place of the platform-specific term “DLL” (Windows).

Moving to the Intel Fortran Compiler

Aspen Plus V8.2 is based on the Intel Fortran compiler XE 2011 (version 12) and Microsoft Visual Studio 2010. This is a change from version V7.3 and earlier versions of Aspen Plus, which used Intel Fortran compiler 9.1 instead, and versions 2004.1 and earlier, which used the no-longer-available Compaq Visual Fortran.

It is possible to continue using Compaq Visual Fortran with Aspen Plus, with certain limitations. See the following section for information on how to configure Aspen Plus to use this compiler, and the text immediately below for details on these limitations.

If you used the Compaq Visual Fortran compiler previously, please be aware of the following issues in upgrading Aspen Plus and/or the Intel Fortran compiler:

- Object files (.OBJ) and static libraries are not compatible between Compaq and Intel compilers. If you move to the Intel compiler you will need to recompile any object files from their source code using the new compiler.
- Use DLLs (dynamic or export libraries) for maximum compatibility between code compiled with different compilers.
- READs and WRITEs to Fortran I/O units opened by code compiled by a different compiler will not work. Aspen Plus opens its pre-defined units (such as the history file and the control panel) using the Intel compiler, so any code compiled by the Compaq compiler will no longer be able to write to these units. See **Compiler-Compatible Write Statements**, below.
- The symbolic debugger will only work if Aspen Plus and the user Fortran code are compiled with the same version of the compiler.

Note: The above issues may apply if you are using Intel Fortran version 11 or earlier, but the READ and WRITE statements will work in this case.

Configuring Aspen Plus for Your Fortran Compiler

The Intel Fortran compiler depends on an external package, such as Microsoft Visual Studio, to provide some capabilities, such as linking. As a result, there are multiple configurations possible on user machines even with the same version of Intel Fortran.

Microsoft's free development tool, Visual C++ Express Edition (2005 or later), includes a linker which will work. The Microsoft Platform SDK is not required for Aspen Plus usage. Installing the latest service pack is recommended.

Information about Visual C++ Express Edition and service packs can be found at <http://msdn.microsoft.com/vstudio/express/downloads/default.aspx>

If you have trouble downloading the product using the web installation, try the manual installation of Express Editions from <http://msdn.microsoft.com/vstudio/express/support/install/>

Aspen Plus provides a utility to allow you to specify the combination of compiler and linker you want to use. This utility sets certain environment variables so that the scripts mentioned in this manual and Aspen Plus will use the tools you specify to compile and link Fortran. The utility runs after your reboot when you first install Aspen Plus. If you need to run it at any other time, you can find it under the **Start** menu under **Programs | AspenTech | Process Modeling <version> | Aspen Plus | Select Compiler for Aspen Plus**.

The utility will present a list of options for different compiler configurations you might be using. In addition, there is a **User** option which allows you to set the environment variables yourself. Below this, it displays the current compiler settings, which come from three sources:

- HKEY_CURRENT_USER, registry settings for the currently logged-in user
- HKEY_LOCAL_MACHINE, registry settings that apply to all users who have not specified user settings
- A value from the Aspen Plus installation that is used if nothing else has been specified.

The utility then allows you to set the options for HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE. Note that you must be an administrator on the computer to set options for HKEY_LOCAL_MACHINE.

If Aspen Plus or an Aspen Plus Simulation Engine window is running when you set these options, your changes may not be applied in the already-running programs. Close them and restart to ensure your new settings are used.

The settings specified by this utility are version-specific and will only affect the version of Aspen Plus under which you ran the utility.

Note: In addition to these settings, users on non-English versions of Windows have reported the need to set the language in Microsoft Visual Studio to **English** instead of **Use Windows default**.

Custom Compiler Settings

If you choose the **User** option in the utility, you will need to set the INCLUDE, LIB, PATH, and USE_COMPAQ_FORTRAN environment variables yourself. The commands in the configuration file listed below illustrate how these should be set.

The list of compiler options displayed by the utility and the actions taken for each can be found in the **Compilers.cfg** file in the **Engine\Xeq** folder of the APrSystem installation. You can add additional configurations to this file, if desired, and the utility will display them as additional options.

Compiler-Compatible Write Statements

WRITE statements writing to Fortran unit numbers for files opened by Aspen Plus will not work correctly if the user Fortran is compiled by a different compiler than Aspen Plus. (You may find the output from these statements in files named fort.## where ## is the unit number.) Instead, you should use the Aspen Plus function DMS_WRTALN to write to these files.

- 1 Include the following (beginning in column 1):

```
#include "ppexec_user.cmn"
```

- 2 Define character string for the buffer. For example, to write a two-line message:

```
CHARACTER*256 BUFFER(2)
```

- 3 Write the two-line message to the buffer:

```
2000 FORMAT ('LINE 1',/, 'LINE 2')  
WRITE (BUFFER, 2000)
```

- 4 Call DMS_WRTALN to write to the history file one line at a time:

```
CALL DMS_WRTALN(USER_NHSTRY, BUFFER(1))  
CALL DMS_WRTALN(USER_NHSTRY, BUFFER(2))
```

Note: Use USER_NRPT rather than USER_NHSTRY to write to the report file.

Example of Writing to History File Using Different Compilers

```
SUBROUTINE USR002  
.  
.  
.  
IMPLICIT REAL*8 (A-H, O-Z)  
#include "ppexec_user.cmn"  
  
DIMENSION ID(2)  
CHARACTER*256 BUFFER(3)  
.  
.  
.  
C  
C WRITE MULTIPLE LINES TO HISTORY FILE  
C  
2000 FORMAT ('FLASH OF STREAM', 2A4, 'FAILED',/,  
+ 'CHECK INPUT SPECIFICATIONS')  
WRITE (BUFFER, 2000) ID(1), ID(2)  
CALL DMS_WRTALN(USER_NHSTRY, BUFFER(1))  
CALL DMS_WRTALN(USER_NHSTRY, BUFFER(2))  
.  
.  
.  
RETURN  
END
```

Writing Fortran User Models

User models written in Fortran should follow these rules and conventions:

Filenames: Files may be given any name, but should end with a **.f** file extension. If you choose to use the **.for** extension, do not use names beginning with an underscore (such as **_abc123.for**), because Aspen Plus may overwrite these files with files containing non-interpretable inline Fortran from models such as Calculator blocks. If you are calling the Aspen Plus

engine directly, do not use the name *runid.for* where *runid* is the Run ID of any of your simulations, because this name will be used for the non-interpretable inline Fortran file in these cases.

Subroutine Names: The names of the physical properties and ADA/PCS user models are dictated by Aspen Plus and should be used as documented in this manual. The names of all other Fortran user models should contain no more than six characters.

Double Precision: All real variables must be declared as double precision (REAL*8). Include the following statement in your user subroutines:

```
IMPLICIT REAL*8 (A-H, O-Z)
```

Aspen Plus Common Blocks: Aspen Plus common blocks are defined in include files. To reference any Aspen Plus common block variables, include the appropriate include file using the C preprocessor syntax. For example, to include common PPEXEC_USER, use the following statement, beginning in column 1:

```
#include "ppexec_user.cmn"
```

The user subroutine should not modify the value of any Aspen Plus common block variables.

Dummy Dimensions: If the Subroutine Argument List Descriptions in this manual show (1) as the Dimension, you should declare that variable as an array with a dummy dimension of 1.

Fortran Extensions: You can use any Fortran extensions supported by your system's compiler, with the exception that subroutine names must not exceed six characters. However, the use of Fortran extensions may make it more difficult to port your user subroutines to other platforms.

Units: All variables in the argument list are in SI units, unless the variable description states otherwise.

Dynamic Linking Overview

Aspen Plus dynamically loads and executes Fortran user models during the run. This feature avoids the need to link special versions of the simulation engine.

Before beginning a run that references Fortran user models, you must:

- Write the user models.
- Compile the user models using the **aspcomp** procedure.
- Link the user models into a Fortran shared library using the **asplink** procedure (optional).
- Supply the object files or shared library to the Aspen Plus system.

During a run, Aspen Plus determines the symbol names of all Fortran user models needed for the run. It then resolves (i.e., finds function pointers to) the user models as follows:

- Loads and resolves symbols from any shared libraries specified via the DLOPT file (see below).

- Looks for additional system-wide customized DLLs in the Engine\Inhouse directory of the APrSystem installation.
- If any symbols remain unresolved, invokes **asplink** in a subprocess to link a new run-specific Fortran shared library from the object module files supplied by the user, then loads and resolves symbols from this new shared library.
- If any symbols remain unresolved, terminates with an error message.

During the dynamic linking process, Aspen Plus writes messages to the file `runid.ld`. This file contains information on objects used in the build and any messages the linker generates and can be used to diagnose any dynamic linking problems.

After resolving all symbols, Aspen Plus invokes the Fortran user models at the appropriate points in the run via a special interface routine named `DMS_DOCALL`. `DMS_DOCALL` is passed the function pointer to the user model of interest, along with all necessary arguments. In turn, it invokes the user model itself, passing it the arguments.

Compiling Fortran User Models

You must compile all Fortran user models before beginning an Aspen Plus run. In order to insure consistent compiler options, use the **aspcomp** procedure for compiling. In an **Aspen Plus Simulation Engine** window (available in **Start | Programs | AspenTech | Process Modeling <version> | Aspen Plus**), type:

```
aspcomp *.f [dbg]
```

The brackets `[]` indicate that the parameter *dbg* is optional. Do not type the brackets. Use *dbg* if you plan to debug these routines.

Supplying Fortran User Models to Aspen Plus

The simplest method of supplying Fortran user models to Aspen Plus is by putting the user model's object module files (the results of the **aspcomp** command) in the run directory. By default, whenever Aspen Plus spawns a subprocess to link a run-specific Fortran shared library it includes all object module files from the run directory.

Alternatively, you can write a Dynamic Linking Options (DLOPT) file that specifies the objects to use when creating the run-specific Fortran shared library. The DLOPT file can also specify shared libraries created by the **asplink** procedure for use when resolving user model symbols instead of, or in addition to, linking a run-specific shared library.

Creating Fortran Shared Libraries Using Asplink

You can use the **asplink** command to create your own Fortran shared libraries containing the object files needed for Fortran user models. This is a good choice when you make many runs without changing your user models. By creating your own shared libraries, you can often avoid the need for Aspen Plus to link a run-specific user model shared library for each run. In an **Aspen Plus Simulation Engine** window (available in **Start | Programs | AspenTech | Process Modeling <version> | Aspen Plus**), type:

```
ASPLINK [DLOPT dloptfile] [genexe] libname
```

Where:

[] Indicates that the contents are optional; do not type the brackets.

dloptfile = Name a DLOPT file.

libname = Name of the Fortran shared library to create. If *libname* ends with .exe or the *genexe* parameter is included, **asplink** builds an EXE instead of a DLL.

Linker messages are written to a file named *libname*.ld.

By default **asplink** includes all of the object module files present in the run directory. This behavior can be modified by specifying a DLOPT file to **asplink**.

The file extension given to the Fortran shared library filename depends on the platform. If you do not specify a file extension in the *libname* parameter to **asplink**, the correct file extension will be attached automatically.

The Fortran shared library file extension for all Windows platforms is **.DLL**.

Writing DLOPT Files

Dynamic Linking Options (DLOPT) files can be used to alter the linking of Fortran shared libraries. DLOPT files can be specified to:

- **asplink** when creating Fortran shared libraries before an Aspen Plus run.
- Aspen Plus when making a run.

DLOPT files can contain:

- DLOPT commands.
- File specifications referring to object module files, object module libraries (archives), or Fortran shared libraries.

Rules for Writing DLOPT Files

Observe these rules when writing DLOPT files:

- Only one DLOPT command or file specification per line.

- File specifications may contain an asterisk (*) as a wildcard for matching a list of files, for example, *.obj.
- File specifications may contain environment variables and UNC paths.
- Do not enclose file specifications in quotation marks (" ").
- Comments may be used anywhere, begin with # or !, and terminate at the end of the line.

DLOPT Commands

The following DLOPT command is recognized:

Command	Platform	Meaning
:no_local	All platforms	Do not use any object module files from the local directory in the link

Example: DLOPT File

```
! This is an example DLOPT file
:no_local                      ! Do not include object module
                              ! files from run directory
D:\USEROBS\*.OBJ              ! Include all object module files from
                              ! D:\USEROBS directory
%USRLIB%\XYZ.LIB              ! Include object module library XYZ.LIB
                              ! from the directory pointed to by the
                              ! USRLIB environment variable
\\SERVER\SHARE\*.DLL          ! Use the shared libraries in the
                              ! \\SERVER\SHARE directory
                              ! when resolving user model symbols
%APRSYS%\Inhouse\*.dll        ! Search the system-level Inhouse
                              ! directory for DLLs.
```

Note: If no DLOPT file is specified for the run, **Aspen Plus** <version>\Engine\xeq\asppfiles.def and **APrSystem** <version>\Engine\xeq\aprsysfiles.def are searched for a DLOPT file specification. A default DLOPT file can be specified in one of these folders, perhaps to reference a system-level set of libraries as in the Inhouse line of the example above.

Specifying DLOPT Files for Aspen Plus Runs

Once a DLOPT file has been written, you can specify it for use during an Aspen Plus run in several ways:

- When you are running the Aspen Plus simulation engine from command line, you can specify the DLOPT file on the command line:

```
aspen input runid /dlopt=dloptfile
```

Where:

input = Input file name
runid = Run id
dloptfile = Name of the DLOPT file

- When you are running Aspen Plus from the user interface, specify the DLOPT file in the **Run Settings** dialog box. From the **Developer** tab of the ribbon, click **Options**. On the **Engine Files** sheet of the dialog box, specify the DLOPT file in the **Linker Options** field.
- When running either from command line or from the user interface, you can specify a DLOPT file in the defaults file. Add the following line to your defaults file:

```
DLOPT:  dloptfile
```

Where:

dloptfile = Name of the DLOPT file (which may include the `${ASPTOP}` variable to refer to the Aspen Plus Engine folder, as in `${ASPTOP}\xeq\DefaultDlopt.opt`)

You can use this method using a defaults file in your own directory. This will cause the named DLOPT file to be used for all runs, unless one of the preceding methods is used to override it.

Note: If you do override the file for a specific run, the default DLOPT file is not used at all. If you override the file but want to also include options from the default file, you will have to copy them into your DLOPT file specific to that run.

- A system administrator can use the above method to configure all users to include the same set of object files in their runs. The Aspen Plus system defaults file, **aspfiles.def**, is located in the XEQ subdirectory of the Aspen Plus simulation engine, and the APrSystem defaults file **aprsysfiles.def** is located in the XEQ subdirectory of the APrSystem engine directory. See **The Aspen Plus Run Definition File** in Chapter 2 of the *Aspen Plus System Management Guide* for more information about this file.

Modifying Asplink

If you need to include any additional Aspen Plus system DLLs in all your asplink runs, you can do so by modifying asplink.prl in the APrSystem `<version>\Engine\xeq` directory. To do so, find this section in asplink.prl:

```
@ap_dlls = ( "atdms", "zemath", "zesqp", "zereport", "ppmon",
"pputil", "ppupp", "zeftools", "zevaraccu", "ppflash", "ppexec",
"zeshs", "pprxn", "ppbase", "ppeos", "zeuosutl", "zestreamu",
"zeitutl", "pppces", "ppstub", "zeusrutl", "pptgs", "atdms2",
"aphier" );
```

Modify it by adding the additional routines you need, such as ppgamma in this example:

```
@ap_dlls = ( "atdms", "zemath", "zesqp", "zereport", "ppmon",
"pputil", "ppupp", "zeftools", "zevaraccu", "ppflash", "ppexec",
"zeshs", "pprxn", "ppbase", "ppeos", "zeuosutl", "zestreamu",
"zeitutl", "pppces", "ppstub", "zeusrutl", "pptgs", "atdms2",
"aphier", "ppgamma" );
```

Using the IMSL Library with Aspen Plus

Some versions of the Intel Fortran come with the VNI/IMSL library. The standard Aspen Plus compiler configuration does not support the use of IMSL library.

In order to use IMSL in Aspen Plus runs, you need to make changes as described here. Note that because of frequent compiler/library changes, you may have to follow the spirit of the steps described here (replace directory paths as appropriate, etc.).

AspenTech thanks our customer Kunle Ogunde from DuPont for sharing some of the information here.

Modify Compilers.cfg to Include IMSL-Related Information

Aspen Plus uses the information in **Compilers.cfg** to set up the compiler environment for compilation and linking for many C/C++/Fortran compiler combinations.

Note: Edit **Compilers.cfg** with a text editor (like NotePad) not a word processor (like Word). Be sure not to introduce any extra line breaks or spaces.

To use the IMSL library, you need to:

- 1 Locate **Compilers.cfg** and make a backup. The file is located in **C:\Program Files\AspenTech\AprSystem <version>\Engine\Xeq**
- 2 Find the appropriate compiler section in **Compilers.cfg** and make the changes.
- 3 Alter **FNL_DIR** if you are using any IMSL library other than version 6.
- 4 Close and reopen any **Aspen Plus Simulation Engine** windows.

The changes required are shown in red in the following example which assumes you are using the IVF10_VS9 configuration.

```
Begin IVF10_VS9 "Intel Fortran 10.x and Microsoft Visual Studio 2008"
IFDir=HKEY_LOCAL_MACHINE("SOFTWARE\Intel\Compilers\Fortran\10#.###\IA32\ProductDir")
VSDir=HKEY_LOCAL_MACHINE("SOFTWARE\Microsoft\VisualStudio\9.0\InstallDir")\..\..
SDKDir=HKEY_LOCAL_MACHINE("SOFTWARE\Microsoft\Microsoft SDKs\Windows\CurrentInstallFolder")
INCLUDE=$(IFDir)\Include;$(VSDir)\vc\atlmfc\include;$(VSDir)\vc\include;$(SDKDir)\include
LIB=$(IFDir)\lib;$(VSDir)\vc\atlmfc\lib;$(VSDir)\vc\lib;$(SDKDir)\lib
PATH=$(IFDir)\bin;$(VSDir)\Common7\IDE;$(VSDir)\vc\bin;$(VSDir)\Common7\Tools;$(SDKDir)\bin;$(PATH)
#
# Start of IMSL changes - non MPI
FNL_DIR=C:\Program Files\VNI\imsl\fnl600
INCLUDE=$(FNL_DIR)\IA32\INCLUDE\dl1;$(INCLUDE)
LIB=$(FNL_DIR)\IA32\LIB;$(LIB)
PATH=$(FNL_DIR)\IA32\LIB;$(PATH)
# End of IMSL changes - non MPI
#
USE_COMPAQ_FORTRAN=
IFDir=
VSDir=
SDKDir=
End
```

Another version of the IMSL library installs in a folder named **CTT6.0** and has a different folder structure. This version needs changes to this section of **Compilers.cfg** like the following:

```
FNL_DIR=C:\Program Files\VNI\CTT6.0
INCLUDE=$(FNL_DIR)\include\IA32;$(INCLUDE)
LIB=$(FNL_DIR)\lib\IA32;$(LIB)
PATH=$(FNL_DIR)\lib\IA32;$(PATH)
```

Modify Fortran Code that Interfaces with IMSL

The AspenPlus compilation script, **aspcomp.bat**, uses the following compiler options:

```
/iface:cvf /MD /Qsave
```

The **/iface:cvf** option specifies the Compaq-Fortran-compatible calling convention.

When you mix object (.obj or .lib) files created using different calling conventions, you must tell the compiler to use the right calling conventions. This is done by adding "CDEC\$ ATTRIBUTES" directives in the Fortran code.

Here is an example of how to properly specify the calling convention when dealing with IMSL routines. This routine itself is compiled with aspcomp.bat.

```
      SUBROUTINE MYSUB(N, X, ...)
      IMPLICIT NONE

c
c      Tells the compiler that the IMSL routine DNEQNF we want to
c      call is compiled with default calling convention
c
cDEC$ ATTRIBUTES DEFAULT :: DNEQNF

c
c      Tells the compiler that the EXTERNAL EQNS to be called
c      by IMSL is compiled with default calling convention
c
cDEC$ ATTRIBUTES DEFAULT :: EQNS

      INTEGER N
      REAL*8  X(N), ....
      EXTERNAL EQNS

      .....

c
c      Calling IMSL DNEQNF routine and pass our EQNS callback
c      routine as EXTERNAL
c
      CALL DNEQNF(EQNS, ERREL, N, ITMAX, XGUESS, X, FNORM)

      .....

      RETURN
      END

      SUBROUTINE EQNS (X, F, N)
```

```

        IMPLICIT NONE
C
C      Tells the compiler to compile EQNS with default calling
C      convention because it will be called by IMSL. Note that
C      the routine that passes EQNS as EXTERNAL to IMSL routine
C      also need the same declaration.
C
CDEC$ ATTRIBUTES DEFAULT :: EQNS
      INTEGER N
      REAL*8  X(N), F(N)

      .....

      RETURN
      END

```

In the example above:

- If you forget to declare DNEQNF, you will get *missing routine _DNEQNF@28* in the linker output (.ld) file, or in general *_ROUTINE@nn* where *ROUTINE* is the IMSL routine missing a declaration and *nn* is 4 times the number of arguments.
- If you forget to declare EQNS (in TWO places), you won't get a linker error but the program will crash when it runs.

Add IMSL Libraries to the Dlopt File for Linking

To use user routines that use the IMSL library in Aspen Plus, or to use asplink to build a DLL that uses the IMSL library, you need to specify IMSL libraries in a dlopt file.

We don't have Intel Fortran with IMSL library and the IMSL document also lacks the information for using IMSL with other products. Our customer Kunle Ogunde from DuPont kindly provided us the following information.

For simple situations, try adding the following lines in the dlopt file:

```

%FNL_DIR%\IA32\lib\imsl.lib
%FNL_DIR%\IA32\lib\IMSLBLAS.LIB
%FNL_DIR%\IA32\lib\IMSL_ERR.LIB
%FNL_DIR%\IA32\lib\libguide40.lib
/NODEFAULTLIB:libcmt.lib

```

If the short list is insufficient, try the following list:

```

%FNL_DIR%\IA32\lib\imsl.lib
%FNL_DIR%\IA32\lib\IMSLSUPERLU.LIB
%FNL_DIR%\IA32\lib\IMSLSCALAR.LIB
%FNL_DIR%\IA32\lib\IMSLBLAS.LIB
%FNL_DIR%\IA32\lib\IMSL_ERR.LIB
%FNL_DIR%\IA32\lib\IMSLMPISTUB.LIB
%FNL_DIR%\IA32\lib\MKL_IA32.lib
%FNL_DIR%\IA32\lib\imslp_err.lib
%FNL_DIR%\IA32\lib\imslsparsestub.lib
%FNL_DIR%\IA32\lib\libguide40.lib
%FNL_DIR%\IA32\lib\mkl_blacs_mpich2.lib
%FNL_DIR%\IA32\lib\mkl_c.lib

```

```
%FNL_DIR%\IA32\lib\mk1_scalapack.lib  
/NODEFAULTLIB:libcmt.lib
```

You may have to do some experiments if you are still getting missing IMSL routines in the linker output (.ld) file.

When using the version of IMSL which installs in the CTT6.0 folder, the folder structure is different, and all these lines need **lib\IA32** rather than **IA32\lib**.

Notes on using asplink.bat to build a DLL that uses the IMSL library

Type "asplink help" for asplink syntax.

The option to specify a dlopt file with asplink is dlopt=dlopt_file (Note: *Not* /dlopt=...).

Typical commands to build MyDll.lib/.dll from myfile1.f and myfile2.f with additional libraries specified in MyDll.opt are as follows

```
aspcomp myfile1.f  
aspcomp myfile2.f  
asplink dlopt=MyDll.opt MyDll  
del myfile1.obj  
del myfile2.obj
```

These commands should be executed from an Aspen Plus Simulation Engine Window.

Be sure to delete loose .obj files because loose .obj files are automatically linked into runid.dll.

If you use object files that call IMSL libraries in an Aspen Plus simulation, then you need to include IMSL libraries in the dlopt file for the Aspen Plus simulation.

If you build MyDll.dll/.lib in advance, and use this prebuilt DLL in your Aspen Plus simulation, then you do not need to include IMSL libraries in the dlopt file for the Aspen Plus simulation, but you *do* need to include MyDll.lib in this dlopt file.

No Recursive Calls

Extensive use of Common blocks in Aspen Plus makes it impossible to safely call subroutines recursively. One execution of the subroutine may alter memory occupied by variables used in the first call, leading to unpredictable results.

2 Calling the Flash Utility

FLSH_FLASH is the Aspen Plus subroutine for all types of flash calculations. A user model can call FLSH_FLASH to perform the following types of flash calculations:

- One-, two-, and three-phase flashes.
- Free-water calculations.
- Flashes with solid substreams.

FLSH_FLASH performs these functions:

- Packs the component vector in each substream.
- Identifies flash types and provides initial guesses.
- Calls flash process routines.
- Calls physical property monitors to compute stream properties.
- Stores results in the stream vector.
- Stores detailed results in COMMON /SHS_STWORK/, COMMON /DMS_STWKWK/, and COMMON /DMS_PLEX/. See Flash Results Stored in COMMON, this chapter. See Appendix A for information on how to include the required common blocks in your code.

After calling FLSH_FLASH, you should check integer function FLSH_FLSTAT to determine the convergence status of the flash.

Flash Utility FLSH_FLASH

Calling Sequence for FLSH_FLASH

```
CALL FLSH_FLASH (SVEC, NSUBS, IDXSUB, ITYPE, NBOPST, KODE,  
NPKODE, KPHASE, MAXIT, TOL, SPEC1, SPEC2, GUESS, LMSG, LPMSG,  
JRES, KRESLT, RETN, IRETN, LCFLAG)
```

Argument List Descriptions for FLSH_FLASH

Variable	I/O [†]	Type	Dimension	Description
SVEC	I/O	REAL*8	(1)	Stream vector (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector 1=MIXED 2=CISOLID 3=NC
NBOPST	I	INTEGER	6	Physical property option set array (see NBOPST)
KODE	I	INTEGER	—	Flash option code (see Flash Types) 1=PQ 2=TP 3=PV 4=TQ 5=TV If NPKODE=1, 1=PQ, 2=TP
NPKODE	I	INTEGER	—	Maximum number of phases in the mixed substream 1=one-phase (phase specified in KPHASE) 2=two-phase (vapor-liquid) 3=three-phase (vapor-liquid-liquid) 11=one-phase with free water (liquid-water) 12=two-phase with free water (vapor-liquid-water)
KPHASE	I	INTEGER	—	Phase when NPKODE=1 1=vapor 2=liquid 3=solid
MAXIT	I	INTEGER	—	Maximum number of iterations. If MAXIT=USER_IUMISS, FLSH_FLASH uses the Maximum Iterations, specified on the Setup Calculation Options Flash Convergence sheet. USER_IUMISS is in COMMON/PPEXEC_USER/.
TOL	I	REAL*8	—	Convergence tolerance If TOL=USER_RUMISS, FLSH_FLASH uses the Error Tolerance specified on the Setup Calculation Options Flash Convergence sheet. USER_RUMISS is in COMMON/PPEXEC_USER/.
SPEC1	I	REAL*8	—	First specified variable (see Flash Types) If KODE=1, SPEC1=P (N/m ²). (P≤0 = pressure drop) If KODE=2, SPEC1=T (K) If KODE=3, SPEC1=P (N/m ²). (P≤0 = pressure drop) If KODE=4, SPEC1=T (K) If KODE=5, SPEC1=T (K)

Variable	I/O [†]	Type	Dimension	Description
SPEC2	I	REAL*8	—	Second specified variable (see Flash Types) If KODE=1, SPEC2=Q (watt) If KODE=2, SPEC2=P (N/m ²). (P ≤ 0 = pressure drop) If KODE=3, SPEC2=V. (vapor/feed molar ratio) If KODE=4, SPEC2=Q (watt) If KODE=5, SPEC2=V
GUESS	I	REAL*8	—	Initial guess (ignored if JRES=2) If KODE=1, GUESS=T If KODE=2, no guess required If KODE=3, GUESS=T If KODE=4, GUESS=P If KODE=5, GUESS=P If GUESS=RMISS, FLSH_FLASH determines the initial guess. USER_RUMISS is in COMMON/PPEXEC_USER/.
LMSG	I	INTEGER	—	Local diagnostic message level
LPMSG	I	INTEGER	—	Local physical property diagnostic level
JRES	I	INTEGER	—	Simulation restart flag 0,1=Do not use retention 2=Use retention
KRESLT	I	INTEGER	—	Result calculation flag 0=Do not calculate results 1=Calculate results -1=Calculate T only (for KODE=1)
RETN	I/O	REAL*8	(1)	Real retention vector (see RETN)
IRETN	I/O	INTEGER	(1)	Integer retention vector (see IRETN)
LCFLAG	O	INTEGER	—	Local convergence flag. Pass this to FLSH_FLSTAT to determine convergence status.

[†]I = Input to subroutine, O = Output from subroutine

Calling Sequence for FLSH_FLSTAT

```

INTEGER FLSH_FLSTAT
...
ISTAT = FLSH_FLSTAT (MODE, LCFLAG)

```

Argument List Descriptions for FLSH_FLSTAT

Variable	I/R [†]	Type	Dimension	Description
MODE	I	INTEGER	—	Specify 0 for this argument. Other values are used internally by Aspen Plus to check for specific types of errors.
LCFLAG	I	INTEGER	—	Value of LCFLAG returned by FLSH_FLASH
ISTAT	R	INTEGER	—	0 = No error 1 = Error 2 = Warning(s), but no error

[†]I = Input to function, R = Value returned by function

Calling Sequence for PPUTL_GOPSET

SUBROUTINE PPUTL_GOPSET (NBOPST, NAME)

Argument List Descriptions for PPUTL_GOPSET

Variable	I/O [†]	Type	Dimensions	Description
NBOPST	O	INTEGER	6	Global property methods array
NAME	O	INTEGER	2	Global property method name

[†]I = Input to subroutine, O = Output from subroutine

NBOPST

Normally NBOPST is provided by Aspen Plus as input to the main user model subroutine. If for some reason this is not available, NBOPST for the global property method may be obtained using PPMON_GOPSET.

Flash Types

Some common flash types are shown in the following table:

Type	KODE	SPEC1	SPEC2
Bubble point	3 (PV) or 5 (TV)	P or T	0
Dew point	3 (PV) or 5 (TV)	P or T	1
Adiabatic	1 (PQ) or 4 (TQ)	P or T	0
Isothermal	2 (TP)	T	P

Specifications for the flash are taken from a combination of values in SVEC, SPEC1, and SPEC2. The component flows and total flow are always used, and the following table shows the additional specifications:

KODE	SPEC1	SPEC2	Additional specifications
1 (PQ)	P	Q	P in SVEC(NCOMP_NCC + 3) if SPEC1 ≤ 0, H in SVEC(NCOMP_NCC + 4)
2 (TP)	T	P	P in SVEC(NCOMP_NCC + 3) if SPEC2 ≤ 0
3 (PV)	P	V	P in SVEC(NCOMP_NCC + 3) if SPEC1 ≤ 0
4 (TQ)	T	Q	H in SVEC(NCOMP_NCC + 4)
5 (TV)	T	V	none

Indices in SVEC are for the MIXED substream. For the CISOLID substream, add NCOMP_NCC + 9 + NCOMP_NVACC. For the NC substream, add NCOMP_NNCC + 9 + NCOMP_NVACC plus an additional NCOMP_NCC + 9 + n if there is a CISOLID substream. See appendix C for details.

RETN

For nonelectrolyte flashes, the current length of RETN is 6*NCC+31. (This length may be changed in the future.) For electrolyte flashes, this length is a function of the number of chemistry reactions. The variable NRETN in COMMON/SHS_STWORK/ contains the actual length needed (see Appendix

A). The real work area for FLSH_FLASH starts at B (STWKWK_LRSTW+1) in COMMON /DMS_PLEX/. This area also contains work space of the correct size for FLSH_FLASH real retention, pointed to by STWORK_MRETN. Thus, the user model can pass B (STWKWK_LRSTW+STWORK_MRETN) to FLSH_FLASH for RETN. If you do this:

- You must set JRES to 0 to turn off restart.
- Retention values are not saved.

IRETN

The current length of IRETN is 2*NCC+24. (This length may be changed in the future.) The integer work area for FLSH_FLASH starts at IB (STWKWK_LISTW+1) in COMMON /DMS_PLEX/. This area also contains work space of the correct size for FLSH_FLASH integer retention, pointed to by STWORK_MIRETN. Thus, the user model can pass IB (STWKWK_LISTW+STWORK_MIRETN) to FLSH_FLASH for IRETN. If you do this:

- You must set JRES to 0 to turn off restart.
- Retention values are not saved.

Flash Results Stored in COMMON

COMMON /DMS_STWKWK/ provides these scalar results from FLSH_FLASH:

Variable Name	Description
STWKWK_TCALC	Temperature (K)
STWKWK_PCALC	Pressure (N/m ²)
STWKWK_VCALC	Vapor fraction (molar)
STWKWK_QCALC	Heat duty (watt)
STWKWK_BETA	Liquid 1/total liquid (molar ratio)
STWKWK_NCPMOO	Number of packed components in the MIXED substream
STWKWK_NCPCSO	Number of packed components among all CISOLID substreams
STWKWK_NCPNCO	Number of packed components among all NC substreams

COMMON /DMS_PLEX/ contains equilibrium compositions in packed form (see Appendix A). These compositions use offsets in:

- COMMON /SHS_STWORK/ (see Appendix A).
- COMMON /DMS_STWKWK/ (offsets STWKWK_LISTW and STWKWK_LRSTW).

Phase	Length	IDX Vector Mole or Mass Fraction Vector	
Overall MIXED substream	STWKWK_NCPMOO	IDX vector: Mole Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIM) B (STWKWK_LRSTW+STWORK_MF)
Liquid	STWKWK_NCPMOO	IDX vector: Mole Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIM) B (STWKWK_LRSTW+ STWORK_MX)
1st liquid	STWKWK_NCPMOO	IDX vector: Mole Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIM) B (STWKWK_LRSTW+ STWORK_MX1)
2nd liquid	STWKWK_NCPMOO	IDX vector: Mole Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIM) B (STWKWK_LRSTW+ STWORK_MX2)
Vapor	STWKWK_NCPMOO	IDX vector: Mole Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIM) B (STWKWK_LRSTW+ STWORK_MY)
Conventional solids	STWKWK_NCPCSO	IDX vector: Mole Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIC) B (STWKWK_LRSTW+ STWORK_MCS)
Nonconventional solids	STWKWK_NCPNCO	IDX vector: Mass Fraction vector:	IB (STWKWK_LISTW+ STWORK_MIN) B (STWKWK_LRSTW+ STWORK_MNC)

FLSH_FLASH packs all CISOLID and NC type substreams into conventional and nonconventional solids arrays, respectively.

3 Calling Physical Property Monitors

You can use Aspen Plus monitor routines, such as PPMON_LMTHMY for liquid mixture thermodynamic properties, to access the Aspen Plus physical property system. Pass the following information to a monitor through the monitor's argument list:

- State variables (temperature, pressure, and composition).
- Calculation codes indicating the required properties.
- Physical property option set pointers.

The composition vector must be in packed form (see Packing Utilities, Chapter 4). A monitor controls calculations of the required properties, using the methods, models, data sets, and model options identified by the option set (property method) pointers. The monitor then returns the properties to the calling program through the argument list.

Aspen Plus provides several thermodynamic phase monitors that can simultaneously control the calculation of:

- Fugacity coefficients.
- Enthalpies.
- Entropies.
- Free energies.
- Molar volumes.

Aspen Plus also provides special monitors with abbreviated argument lists for individual properties, such as PPMON_ENTHL for liquid mixture enthalpy. When possible, the phase monitors avoid redundant calculations, such as when an equation of state is used to calculate several properties of a given mixture. Use phase monitors when appropriate to increase computational efficiency. Avoid calling several individual property monitors.

Aspen Plus can also calculate property derivatives analytically. The calculated derivatives include derivatives with respect to temperature (T), pressure (P), mole number (n), and mole fraction (x). Aspen Plus provides a general property monitor PPMON_CALPRP for calculation of thermodynamic and transport properties and their derivatives.

This chapter lists the monitors available for thermodynamic, transport, and nonconventional properties, along with their argument lists.

Calling Sequences for Thermodynamic Property Monitors

Pure Component Thermodynamic Phase Monitors (Vapor)

```
CALL PPMON_VTHRM (T, P, N, IDX, NBOPST, KDIAG, KBASE,  
                  KPHI, KH, KS, KG, KV, PHI, H, S, G, V,  
                  DPHI, DH, DS, DG, DV, KER)
```

Pure Component Thermodynamic Phase Monitors (Liquid)

```
CALL PPMON_LTHRM (T, P, N, IDX, NBOPST, KDIAG, KBASE,  
                  KPHI, KH, KS, KG, KV, PHI, H, S, G, V,  
                  DPHI, DH, DS, DG, DV, KER)
```

Pure Component Thermodynamic Phase Monitors (Solid)

```
CALL PPMON_STHRM (T, P, N, IDX, NBOPST, KDIAG, KBASE,  
                  KPHI, KH, KS, KG, KV, PHI, H, S, G, V,  
                  DPHI, DH, DS, DG, DV, KER)
```

Mixture Thermodynamic Phase Monitors (Vapor)

```
CALL PPMON_VMTHRM (T, P, Y, N, IDX, NBOPST, KDIAG, KBASE,  
                  KPHI, KH, KS, KG, KV, PHI, HMX, SMX, GMX,  
                  VMX, DPHI, DHMX, DSMX, DGMX, DVMX, KER)
```

Mixture Thermodynamic Phase Monitors (Liquid)

```
CALL PPMON_LMTHMY (T, P, X, Y, N, IDX, NBOPST, KDIAG,  
                  KBASE, KPHI, KH, KS, KG, KV, PHI, HMX,  
                  SMX, GMX, VMX, DPHI, DHMX, DSMX, DGMX,  
                  DVMX, KER)
```

Mixture Thermodynamic Phase Monitors (Solid)

```
CALL PPMON_SMTHRM (T, P, CS, N, IDX, NBOPST, KDIAG, KBASE,  
                  KPHI, KH, KS, KG, KV, PHI, HMX, SMX, GMX,  
                  VMX, DPHI, DHMX, DSMX, DGMX, DVMX, KER)
```

Equilibrium Ratio (K-Value) (Vapor-Liquid)

```
CALL PPMON_KVL (T, P, X, Y, N, IDX, NBOPST, KDIAG, KK, K,  
                DK, KER)
```

Equilibrium Ratio (K-Value) (Liquid-Liquid)

```
CALL PPMON_KLL (T, P, X1, X2, N, IDX, NBOPST, KDIAG,  
                KK, K, DK, KER)
```

Mixture Enthalpy (Vapor)

```
CALL PPMON_ENTHV (T, P, Y, N, IDX, NBOPST, KDIAG, KBASE, KH,  
HMX, HMX, KER)
```

Mixture Enthalpy (Liquid)

```
CALL PPMON_ENTHL (T, P, X, N, IDX, NBOPST, KDIAG, KBASE, KH,  
HMX, DHMX, KER)
```

Mixture Enthalpy (Solid)

```
CALL PPMON_ENTHS (T, P, CS, N, IDX, NBOPST, KDIAG, KBASE, KH,  
HMX, DHMX, KER)
```

Mixture Molar Volume (Vapor)

```
CALL PPMON_VOLV (T, P, Y, N, IDX, NBOPST, KDIAG, KV,  
VMX, DVMX, KER)
```

Mixture Molar Volume (Liquid)

```
CALL PPMON_VOLL (T, P, X, N, IDX, NBOPST, KDIAG, KV,  
VMX, DVMX, KER)
```

Mixture Molar Volume (Solid)

```
CALL PPMON_VOLS (T, P, CS, N, IDX, NBOPST, KDIAG, KV,  
VMX, DVMX, KER)
```

Mixture Fugacity Coefficient (Vapor)

```
CALL PPMON_FUGV (T, P, Y, N, IDX, NBOPST, KDIAG, KPHI,  
PHI, DPHI, KER)
```

Mixture Fugacity Coefficient (Liquid)

```
CALL PPMON_FUGLY (T, P, X, Y, N, IDX, NBOPST, KDIAG,  
KPHI, PHI, DPHI, KER)
```

Mixture Fugacity Coefficient (Solid)

```
CALL PPMON_FUGS (T, P, CS, N, IDX, NBOPST, KDIAG, KPHI, PHI,  
DPHI, KER)
```

Ideal Gas

```
CALL PPBASE_IDLGAS (T, Y, N, IDX, KDIAG, KBASE, KHI, KSI,  
KGI, KH, KS, KG, H, S, G, DH, DS, DG,  
HMX, SMX, GMX, DHMX, DSMX, DGMX, KER)
```

Calling Sequences for Transport Property Monitors

Mixture Viscosity (Vapor)

```
CALL PPMON_VISCV (T, P, Y, N, IDX, NBOPST, KDIAG, VISC,  
KER)
```

Mixture Viscosity (Liquid)

```
CALL PPMON_VISCL (T, P, X, N, IDX, NBOPST, KDIAG, VISC,  
KER)
```

Mixture Thermal Conductivity (Vapor)

```
CALL PPMON_TCONV (T, P, Y, N, IDX, NBOPST, KDIAG, TCON,  
KER)
```

Mixture Thermal Conductivity (Liquid)

```
CALL PPMON_TCONL (T, P, X, N, IDX, NBOPST, KDIAG, TCON,  
KER)
```

Mixture Thermal Conductivity (Solid)

```
CALL PPMON_TCONS (T, P, CS, N, IDX, NBOPST, KDIAG, TCON, KER)
```

Mixture Diffusion Coefficient (Vapor)

```
CALL PPMON_DIFCOV (T, P, Y, N, IDX, NBOPST, KDIAG, DIFCO, KER)
```

Mixture Diffusion Coefficient (Liquid)

```
CALL PPMON_DIFCOL (T, P, X, N, IDX, NBOPST, KDIAG, DIFCO, KER)
```

Binary Diffusion Coefficient (Vapor)

```
CALL PPMON_DFCOBV (T, P, Y, N, IDX, NBOPST, KDIAG, BINCO, KER)
```

Binary Diffusion Coefficient (Liquid)

```
CALL PPMON_DFCOBL (T, P, X, N, IDX, NBOPST, KDIAG, BINCO, KER)
```

Mixture Surface Tension

```
CALL PPMON_SRFTNY (T, P, X, Y, N, IDX, NBOPST, KDIAG, SRFTEN,  
KER)
```


Calling Sequences for Nonconventional Property Monitors

Nonconventional Enthalpy Monitor

```
CALL PPNCS_ENTHAL (IDXNC, CAT, T, P, KDIAG, KNC, HNC,  
                  DHNC, KER)
```

Nonconventional Density Monitor

```
CALL PPNCS_DENSTY (IDXNC, CAT, T, P, KDIAG, KNC, DNC,  
                  DDNC, KER)
```

Calling Sequences for Thermodynamic and Transport Property Monitors with Derivatives

General Property Monitor

```
CALL PPMON_CALPRP (T, P, Z, NX, N, IDX, NBOPST, KDIAG,  
                  KBASE, PROPS, NPROP, PHASES, NPHASE,  
                  RESULT, NRESULT, IRESULT, KERR)
```

Note: To avoid excessive memory usage in simulations with many components, you may not request both dx (mole fraction) and dn (mole number) derivatives in a single call to PPMON_CALPRP.

Argument Descriptions for Physical Property Monitors

Argument List Descriptions for Physical Property Monitors

Variable	I/O [†]	Type	Dimension	Description
T	I	REAL*8	—	Temperature (K)
P	I	REAL*8	—	Pressure (N/m ²)
NX	I	INTEGER	—	Number of phases in the composition vector Z
N	I	INTEGER	—	Number of components present
IDX	I	INTEGER	N	Component index vector (see IDX)

Variable	I/O[†]	Type	Dimension	Description
IDXNC	I	INTEGER	—	Nonconventional component index (see IDXNC)
Y	I	REAL*8	N	Vapor mole fraction vector (see Y, X, X1, X2, Z, CS)
X	I	REAL*8	N	Liquid mole fraction vector (see Y, X, X1, X2, Z, CS)
Z	I	REAL*8	N*NX	Mole fraction matrix for all phases for PPMON_CALPRP (see Y, X, X1, X2, Z, CS)
X1	I	REAL*8	N	Liquid 1 mole fraction vector (see Y, X, X1, X2, Z, CS)
X2	I	REAL*8	N	Liquid 2 mole fraction vector (see Y, X, X1, X2, Z, CS)
CS	I	REAL*8	N	Conventional solid mole fraction vector (see Y, X, X1, X2, CS)
CAT	I	REAL*8	(1)	Component attribute vector (see CAT)
NBOPST	I	INTEGER	6	Physical property option set vector (see NBOPST)
KDIAG	I	INTEGER	—	Diagnostic level code (see KDIAG)
KBASE	I	INTEGER	—	Thermodynamic function base code. KBASE=0, Pure components in ideal gas state at 298.15 K; KBASE=1, elements in their standard states at 298.15 K.
KK	I	INTEGER	—	K-value calculation code (see Calculation Codes)
KPHI	I	INTEGER	—	Fugacity coefficient calculation code (see Calculation Codes)
KH	I	INTEGER	—	Enthalpy calculation code (see Calculation Codes)
KS	I	INTEGER	—	Entropy calculation code (see Calculation Codes)
KG	I	INTEGER	—	Gibbs energy calculation code (see Calculation Codes)
KV	I	INTEGER	—	Molar volume calculation code (see Calculation Codes)
KHI	I	INTEGER	—	Pure component ideal gas enthalpy calculation code (see Calculation Codes)
KSI	I	INTEGER	—	Pure component ideal gas entropy calculation code (see Calculation Codes)
KGI	I	INTEGER	—	Pure component ideal gas Gibbs energy calculation code (see Calculation Codes)
KNC	I	INTEGER	—	Nonconventional property calculation code (see Calculation Codes)
PROPS	I	CHARACTER	NPROP	Vector of properties. Properties are calculated for all phases. (see Derivatives)
NPROP	I	INTEGER	—	Number of properties in PROPS
PHASES	I	CHARACTER	NPHASE	Vector of phases for which properties are to be calculated (see Phases)
NPHASE	I	INTEGER	—	Number of phases in PHASES (see Phases)

Variable	I/O [†]	Type	Dimension	Description
RESULT	O	REAL*8	*	Calculated properties. (see CALPRP Results)
NRESULT	I/O	INTEGER	—	Number of elements in RESULT. (see CALPRP Results)
IRESULT	O	INTEGER	NPROP* NPHASE	Index into RESULT for each property in each phase. (see CALPRP Results)
PHI	O	REAL*8	N	Vector of fugacity coefficients of pure components or of components in a mixture
DPHI	O	REAL*8	N	Vector of partial derivatives with respect to temperature of fugacity coefficients of pure components or of components in a mixture (K^{-1})
K	O	REAL*8	N	Vector of equilibrium ratios (K-values) of components in coexisting phases
DK	O	REAL*8	N	Vector of partial derivatives of equilibrium ratios (K-values) of components in coexisting phases with respect to temperature
H	O	REAL*8	N	Vector of pure component enthalpies (J/kgmole)
DH	O	REAL*8	N	Vector of partial derivatives of pure component enthalpies with respect to temperature (J/kgmole-K)
HMX	O	REAL*8	—	Mixture enthalpy (J/kgmole)
DHMX	O	REAL*8	—	Partial derivative of mixture enthalpy with respect to temperature (J/kgmole-K)
S	O	REAL*8	N	Vector of pure component entropies (J/kgmole-K)
DS	O	REAL*8	N	Vector of partial derivatives of pure component entropies with respect to temperature (J/kgmole-K ²)
SMX	O	REAL*8	—	Mixture entropy (J/kgmole-K)
DSMX	O	REAL*8	—	Partial derivative of mixture entropy with respect to temperature (J/kgmole-K ²)
G	O	REAL*8	N	Vector of pure component Gibbs energies (J/kgmole)
DG	O	REAL*8	N	Vector of partial derivatives of pure component Gibbs energies with respect to temperature (J/kgmole-K)
GMX	O	REAL*8	—	Mixture Gibbs energy (J/kgmole)
DGMX	O	REAL*8	—	Partial derivative of mixture Gibbs energy with r with respect to temperature (J/kgmole-K)
V	O	REAL*8	N	Vector of pure component molar volumes (m ³ /kgmole)
DV	O	REAL*8	N	Vector of partial derivatives of pure component molar volumes with respect to temperature (m ³ /kgmole-K)
VMX	O	REAL*8	—	Mixture molar volume (m ³ /kgmole)
DVMX	O	REAL*8	—	Partial derivative of mixture molar volume with respect to temperature (m ³ /kgmole-K)

Variable	I/O [†]	Type	Dimension	Description
VISC	O	REAL*8	—	Mixture viscosity (N-s/m ²)
TCON	O	REAL*8	—	Mixture thermal conductivity (J/sec-m-K)
DIFCO	O	REAL*8	N	Vector of diffusion coefficients of components in a mixture (m ² /sec)
BINCO	O	REAL*8	N*N	Matrix of binary diffusion coefficients of components (m ² /sec). BINCO(I,J)=BINCO(J,I).
SRFTEN	O	REAL*8	—	Mixture surface tension (N/m)
HNC	O	REAL*8	—	Enthalpy of a nonconventional component (J/kg)
DHNC	O	REAL*8	—	Partial derivative of the enthalpy of a nonconventional component with respect to temperature (J/kg-K)
DNC	O	REAL*8	—	Density of a nonconventional component (m ³ /kg)
DDNC	O	REAL*8	—	Partial derivative of the density of a nonconventional component with respect to temperature (m ³ /kg-K)
KER	O	INTEGER	—	Error return code ($\neq 0$ if an error or warning condition occurred in any physical property model; $= 0$ otherwise)
KERR	O	INTEGER	—	Error flag for CALPRP. (see CALPRP Results)

[†]I = Input to subroutine, O = Output from subroutine

IDX

IDX is a vector containing component sequence numbers of the conventional components actually present in the order that they appear on the Components Specifications Selection sheet. For example, if only the first and third conventional components are present, N=2 and IDX=(1,3). You can use the utility routine SHS_CPACK to pack a stream and produce an IDX vector. (See Chapter 4 for a description of SHS_CPACK.)

IDXNC

IDXNC is the index of the single nonconventional component for which calculations are to be performed. IDXNC is the sequence number of the component within the complete set of attributed components, both conventional and nonconventional, in the simulation.

The order for the set of attributed components is the order on the Components Specifications Selection sheet.

Y, X, X1, X2, Z, CS

Y, X, X1, X2, and CS are packed mole fraction vectors of length N. They contain the mole fractions of the components that are present and are to be included in the calculations. The corresponding IDX vector defines the order of components. For example, if N=2 and IDX=(1,3), Y(2) is the vapor mole

fraction of the component listed third on the Components Specifications Selection sheet, excluding any nonconventional components.

For PPMON_CALPRP, Z is a packed mole fraction matrix of dimension N by NX. The first N elements correspond to phase 1, the next N to phase 2, and so on. The identity of each phase (e.g. liquid or vapor) is determined by the PHASES specification.

CAT

CAT is the vector of component attribute values for the complete set of attributed nonconventional components in the simulation. For a given nonconventional component, the index IDXNC and additional indices described in Chapter 6 provide sufficient information for a monitor to retrieve attribute values from the CAT vector.

NBOPST

Normally NBOPST is provided by Aspen Plus as input to the main user model subroutine. If for some reason this is not available, NBOPST for the global property method may be obtained using PPMON_GOPSET.

Calling Sequence for PPUTL_GOPSET

```
SUBROUTINE PPUTL_GOPSET (NBOPST, NAME)
```

Argument List Descriptions for PPUTL_GOPSET

Variable	I/O [†]	Type	Dimensions	Description
NBOPST	O	INTEGER	6	Global property methods array
NAME	O	INTEGER	2	Global property method name

[†]I = Input to subroutine, O = Output from subroutine

KDIAG

KDIAG controls the printing of error, warning, and diagnostic information in the history file. KDIAG values correspond to the values for message levels described in the help for the message level fields on the **Setup | Specifications | Diagnostics** sheet or the **Block Options | Diagnostics** sheet of a unit operation model. If KDIAG=N, all messages at level N and below are printed.

Calculation Codes

Calculation codes can have four values:

- 0 = Do not calculate property or its temperature derivative
- 1 = Calculate the property only
- 2 = Calculate the temperature derivative of the property only
- 3 = Calculate the property and its temperature derivative

The monitors do not change values of argument list quantities that will not be calculated.

Phases

Valid phase specifiers for PPMON_CALPRP are:

V= Vapor
L= Liquid
S= Solid

NPHASE is generally the same as NX.

CALPRP Results

The results are arranged in this order:

- All properties for phase 1.
- All properties for phase 2.
- ...
- All properties for phase NPHASE.

NRESULT should be set to the length of the RESULT vector provided. If NRESULT is 0, no calculations are performed and NRESULT is set to the actual size required; user should allocate required storage and call the PPMON_CALPRP property monitor again. If NRESULT is insufficient, results are truncated and an error code of -3 is returned.

IRESULT provides indices into RESULT for each property in each phase. IRESULT allows you to easily retrieve the desired property in each phase.

KERR error codes for PPMON_CALPRP are:

- | | |
|----|--|
| 0 | No error. |
| -1 | Error in parsing property specifications. |
| -2 | Invalid phase specification. |
| -3 | Insufficient size for result vector. |
| -4 | Insufficient size of mole fraction vector. |

Derivatives

To specify the derivative of a property, append one of the following suffixes to the property name in PROPS:

- .DT for derivative with respect to temperature
- .DP for derivative with respect to pressure
- .DN for derivative with respect to mole number
- .DX for derivative with respect to mole fraction

For example, to calculate the temperature derivative of mixture enthalpy, specify HMX.DT in PROPS.

To avoid excessive memory usage in simulations with many components, you may not request both DX (mole fraction) and DN (mole number) derivatives in a single call to PPMON_CALPRP.

Calling Sequences for PROP-SET Property Monitors

General PROP-SET Property Monitor (CALUPP)

```
CALL UPP_CALUPP (T, P, X, N, IDX, NBOPST, IAS, PROPS, PHASES,  
                KWDBS, XPCLV, KULAB, NL2C, KL2C, NATOMS,  
                KATOMS, NPKODE, RESULT, KERR)
```

This monitor is used to calculate the prop-set properties listed in Chapter 4 of the *Physical Property Data* Reference Manual.

Note: CALUPP cannot be used for properties based on flow rate. For properties like VLSTDMX which can be returned in either flow-based units or units per mole or mass, set KULAB to the appropriate mole or mass units and multiply the result from CALUPP by the flow rate to get the desired value.

Note: CALUPP should not be called from user property subroutines. Doing so may result in recursive function calls which may overwrite variables (such as those in Common blocks) used in the first call, which that call may still need. This will lead to unpredictable results.

PROP-SET Property Monitor (CALUP1)

```
CALL UPP_CALUP1 (PROPS, LDT, LDP, LDN, KERR)
```

This utility can be used to query the dependency of the given property on temperature, pressure and composition.

PROP-SET Property Monitor (CALUP2)

```
CALL UPP_CALUP2 (IDPSET, NPROP, PROPS, NTEMP, TEMP, NPRES,  
                PRES, NPHASE, PHASES, NCOMP, IDCOMP, NBASIS,  
                KWDBS, NUNIT, KULAB, NLVPCT, XPCLV, NL2C,  
                KL2C, KERR)
```

This utility can be used to query the property names, phase, basis, unit, and other qualifiers for the given Prop-Set ID.

PROP-SET Property Monitor (CALUP3)

```
CALL UPP_CALUP3 (T, P, X, N, IDX, NBOPST, IAS, PROPS, LP,  
                LDT, LDP, LDN, PHASES, KWDBS, XPCLV, KULAB,  
                NL2C, KL2C, NATOMS, KATOMS, NPKODE, RESULT,  
                RESUDT, RESUDP, RESUDN, KERR)
```

This monitor is used to calculate the property and derivatives of the prop-set properties listed in **Property Sets** in *Physical Property Data* in online help. This monitor is similar to CALUPP.

Argument Descriptions for PROP-SET Property Monitors

Argument List Descriptions for PROP-SET Property Monitors

Variable	I/O [†]	Type	Dimension	Description
T	I	REAL*8	—	Temperature (K)
P	I	REAL*8	—	Pressure (Pa)
X	I	REAL*8	N	Composition vector
N	I	INTEGER	—	Number of components
IDX	I	INTEGER	N	Vector of component index
NBOPST	I	INTEGER	6	Physical property option set vector
IAS	I	INTEGER	—	ID of entry ASSAY for petroleum properties
PROPS	I/O	INTEGER (2,*)		Matrix of property which is calculated (see PROPS)
PHASES	I/O	INTEGER *		Vector of index of phase for which property to be calculated (see PHASES)
KWDBS	I/O	INTEGER *		Vector of basis index (see KWDBS)
XPCLV	I/O	REAL *8 *		Vector of Liquid volume percent % (see XPCLV)
KULAB	I/O	INTEGER (4,*)		Matrix of unit output of result (see KULAB)
KL2C	I/O	INTEGER NL2C		Vector of key component index of the second liquid phase
NL2C	I/O	INTEGER	—	Number of component index in KL2C
NATOMS	I	INTEGER	—	Number of atom entries in KATOMS
KATOMS	I	INTEGER NATOMS		Vector of atom entries
LP	I	LOGICAL	—	.TRUE. : Property required. .FALSE. : No calculation
LDT	I/O	LOGICAL	—	.TRUE. : DT required. .FALSE. : No calculation
LDP	I/O	LOGICAL	—	.TRUE. : DP required. .FALSE. : No calculation
LDN	I/O	LOGICAL	—	.TRUE. : DN required. .FALSE. : No calculation
NPROP	O	INTEGER	—	Number of properties
NPHASE	O	INTEGER	—	Number of phases
NBASIS	O	INTEGER	—	Number of basis
NUNIT	O	INTEGER	—	Number of units
NLVPCT	O	INTEGER	—	Number of liquid volume %
NCOMP	O	INTEGER	—	Number of properties
IDCOMP	O	INTEGER NCOMP		Vector of ID of components
NTMP	O	INTEGER	—	Number of temperature points
TEMP	O	REAL*8	NTMP	Vector of temperature
NPRES	O	INTEGER	—	Number of pressure points

Variable	I/O [†]	Type	Dimension	Description
PRES	O	REAL*8	NPRES	Vector of pressure
RESULT	O	REAL*8	*	Calculated property (see CALUPP results)
RESUDT	O	REAL*8	*	Calculated DT (see CALUPP results)
RESUDP	O	REAL*8	*	Calculated DP (see CALUPP results)
RESUDN	O	REAL*8	*	Calculated DN (see CALUPP results)
KERR	O	INTEGER	—	Error flag for CALUPP (see KERR)

[†]I = Input to subroutine, O = Output from subroutine

PROPS

PROPS is an integer vector containing the name of the requested prop-set property. Valid property names are listed in **Property Sets** in *Physical Property Data* in online help. In addition to these, names of user properties (defined on the **Customize | User Properties** form in the **Properties** environment) can also be used with CALUPP.

For example, to request pure-component heat capacity CP:

```
INTEGER PROPS(2)
DATA PROPS /4HCP      ,4H      /
```

PHASES

PHASES is the index of the phase qualifier (see **Property Sets** in *Physical Property Data* in online help).

- 1 V (vapor)
- 2 L (liquid phase)
- 3 S (solid phase)
- 4 L1 (first liquid phase)
- 5 T (total mixture for mixed substream)
- 6 L2 (second liquid phase)

For example, to request a property for the vapor phase:

```
INTEGER PHASES
DATA PHASES / 1 /
```

KWDBS

KWDBS is the index of the WET/DRY basis qualifier (Default is WET)

- 1 WET (to include water in the calculation)
- 2 DRY (to exclude water from the calculation)

For example, to specify DRY basis for a property calculation:

```
INTEGER KWDBS
DATA KWDBS / 2 /
```

XPCLV

XPCLV is the liquid volume percent (0-100), which is used only for TBPT, D86T, D1160T, VACT, D86T-API5, TBPTWT, D86TWT, D1160TWT, VACTWT, D86TWT-API, D2887T, D86TCK, D86TWTCK. For other properties, set XPCLV to RGLOB_RMISS (from COMMON/DMS_RGLOB/).

KULAB

KULAB is an integer vector containing the UNITS qualifier for the calculated property (result). For default, OUT-UNITS is to be used.

For example, to request mixture enthalpy in Btu/hr:

```
INTEGER KULAB(4)
DATA KULAB /4HBTU /,4HHR ,4H ,4H /
```

To use the default, OUT-UNITS, set KULAB = 0 or blank:

```
INTEGER KULAB(4)
DATA KULAB / 4H ,4H ,4H ,4H /
```

To specify a units set, such as SI, for the results, set KULAB to the units set name:

```
INTEGER KULAB(4)
DATA KULAB / 4HSI ,4H ,4H ,4H /
```

Note: CALUPP cannot be used for properties based on flow rate. For properties like VLSTDMX which can be returned in either flow-based units or units per mole or mass, set KULAB to the appropriate mole or mass units and multiply the result from CALUPP by the flow rate to get the desired value.

CALUPP Results

CALUPP can only handle one property for one phase, one basis, one liquid volume percent and one unit label in the same call. If you need several properties for multiple phases, basis and so on, you must make multiple calls to the property monitor.

You must define (dimension) the length of the result vectors (RESULT, RESUDT, RESUDP and RESUDN) to be of sufficient size. For mixture properties, RESULT has length 1. For pure component and partial properties, RESULT has length N. Temperature and pressure derivative require the same storage area as the property itself. Mole number derivative requires storage area equal to N times the area required for the property.

KERR

Error codes for UPP_CALUPP are:

- | | |
|----|---|
| 0 | No error; the property is to be calculated |
| >0 | Error in processing property set; no calculation |
| 1 | Error for creating prop-set |
| 2 | No property requested |
| 3 | Dry basis is invalid because no water in the component list |
| 4 | No component in composition vector |

5 Liquid volume percent value is outside
>6 Flash failure

Example of Calling CALUPP Multiple Times to Retrieve Multiple Properties

```
      SUBROUTINE SAMPLE (T, P, X, N, IDX, NBOPST, CPV, CPL,  
        + CPVMX, CPLMX, HV, HL, HVMX, HLMX)  
C  
C Sample calculations of 2 properties for both pure components  
C and mixture for 2 phases (vapor and liquid).  
C  
C CALUPP can only handle one property and one phase per call, so  
C it must be called 4 times, 1 for each combination of property  
C and phase. This example collects the data for these calls in  
C arrays and passes the appropriate parts for each call.  
C  
C T, P, X, N, IDX, and NBOPST are supplied through arguments  
C in this subroutine. NBOPST is provided as input to most user  
C subroutines. See IDX and NBOPST in this chapter for  
C assistance in determining these values.  
C  
C T -- temperature  
C P -- pressure  
C X -- component vector  
C N -- number of components  
C IDX -- component index vector  
C NBOPST -- option set vector  
C  
      IMPLICIT NONE  
  
      INTEGER NBOPST(6), N, IDX(N)  
      REAL*8 T, P, X(N), CPV(N), CPL(N), CPVMX, CPLMX, HV(N),  
        + HL(N), HVMX, HLMX  
C  
C Set qualified properties  
C NPROP -- number of qualified properties  
C PROPS(2,NPROP) -- names of qualified properties  
C  
C Note: To retrieve both pure and mixture properties, the  
C number of properties is two for each property. For example,  
C heat capacity is CP for pure component, CPMX for mixture. So  
C for this case the number of properties (NPROP) is four:  
C CP, CPMX, H, HMX.  
C  
      INTEGER NPROP  
      PARAMETER (NPROP = 4)  
      INTEGER PROPS(2,NPROP)  
      DATA PROPS /4HCP ,4H ,4HCPMX,4H , 4HH ,4H ,4HHMX ,4H /  
C  
C Set qualified phases
```

```

C NPHASE -- number of qualified phases
C For each property:
C PHASES(NPHASE,NPROP) -- index of qualified phases for each
C phase of each property
C 1: vapor
C 2: liquid
C 3: solid
C 4: liquid 2
C 5: total
C 6: liquid 1
C
      INTEGER NPHASE
      PARAMETER (NPHASE = 2)
      INTEGER PHASES(NPHASE,NPROP)
      DATA PHASES / 1, 2, 1, 2, 1, 2, 1, 2 /
C
C Set qualified wet/dry basis
C KWDBS(NPROP) -- index of basis for each property
C 1 : wet
C 2 : dry
C Note: For all pure component properties,
C electrolyte properties (Phys. Prop. Data table 4.4), and
C nonconventional properties (Phys. Prop. Data table 4.8),
C wet/dry basis is not valid. Set KWDBS = 0 in these cases.
C For other properties, if this is not specified
C (that is, if KWDBS = 0), then wet is the default.
C In this case, KWDBS = 0 for pure component properties (CP, H)
C KWDBS = 1 for mixture properties (CPMX, HMX)
C
      INTEGER KWDBS(NPROP)
      DATA KWDBS / 0, 1, 0, 1 /
C
C Set qualified liquid volume percent
C XPCLV -- values of liquid volume %
C Note: Most properties don't need liquid volume percent.
C For them, set XPCLV = RMISS (1D35)
C For the following properties, liquid volume percent must be
C supplied.
C TBPT, D86T, D1160T, VACT, D86T-API5, TBPTWT,
C D86TWT, D1160TWT, VACTWT, D86TWT-API, D2887T,
C D86TCK, D86TWTCK
C In this case, for all properties XPCLV is not valid, so
C set all XPCLV = RMISS (1D35).
C If you needed different percent for different properties you
C would create an array as for other parameters above.
C
      REAL*8 XPCLV
      DATA XPCLV / 1D35 /
C
C
C Set qualified units output of the result
C KULAB(4,NPROP) -- units output for each property
C Note: If this is not specified, the default is the in-units
C specified in Aspen Plus or Aspen Properties
C
      INTEGER KULAB(4,NPROP)
      DATA KULAB / 4HJ/KM, 4HOL-K,4H , 4H ,

```

```

      + 4H , 4H ,4H , 4H ,
      + 4HJ/KM, 4HOL ,4H , 4H ,
      + 4H , 4H ,4H , 4H /
C
C Set L2-comps
C NL2C -- Number of L2-comps
C KL2C(NL2C) -- IDs of L2-comps
C For this case there are no L2-comps required, so set NL2C = 0
C If you needed different L2-comps for different properties you
C would create an array as for other parameters above.
C
      INTEGER NL2C
      INTEGER KL2C
      DATA NL2C / 0 /
      DATA KL2C / 0 /
C
C Set atoms entries
C NATOMS -- number of atoms entries
C KATOMS(NATOMS) -- atom vector entries
C For this case there are no properties requiring atoms entries,
C so set NATOMS = 0
C If you needed different atoms for different properties you
C would create an array as for other parameters above.
C
      INTEGER NATOMS
      INTEGER KATOMS
      DATA NATOMS / 0 /
      DATA KATOMS / 0 /
C
C Set number of phases for flash
C NPKODE -- number of phases for flash
C 1: 1-phase
C 2: 2-phases
C 3: 3-phases
C Note: If this is not specified (NPKODE = 0), a 2-phase flash is
C performed.
C If you needed different numbers of phases for different
C properties you would create an array as for other parameters
C above.
C
      INTEGER NPKODE
      DATA NPKODE / 0 /
C
C Set result vector
C RESULT(?) -- returned values of properties
C Note: You must set the maximum length of the result so that it
C is large enough.
C * For most mixture properties, result has length = 1
C * For pure-component or partial properties, length = N
C * For a petroleum property curve, length = 2*NPOINT
C   1 to NPOINT --> liq. vol. % for each point
C   NPOINT+1 to 2*NPOINT --> petroleum property
C
      REAL*8 RESULT(N)
C
C Set order number of entry assay for calculations of petroleum
C properties

```

```

C IAS > 0 for petroleum property calculations
C IAS = 0 for others
C In this case, no petroleum property is requested. Set IAS = 0.
C If you needed different assay numbers for different properties
C you would create an array as for other parameters above.
C
      INTEGER IAS
      DATA IAS / 0 /
C
C Set local variables
C
      INTEGER KERR, I, II, J, JJ, K, KK
C
      KERR = 0
C
C Calculate each property for each phase
C Note how corresponding elements of each array are passed to
C CALUPP for each call for a particular property and phase.
C
      DO 10 I = 1, NPROP
        DO 20 J = 1, NPHASE
          CALL UPP_CALUPP(T, P, X, N, IDX, NBOPST, IAS,
1             PROPS(1,I), PHASES(J,I), KWDBS(I), XPCLV,
2             KULAB(1,I), NL2C, KL2C, NATOMS, KATOMS, NPKODE,
3             RESULT, KERR)

          IF (KERR .NE. 0) GO TO 20
C
C Copy results into appropriate vectors
C
          IF (I.EQ.2 .OR. I.EQ.4) THEN
            IF (J .EQ. 1) THEN
              IF (I .EQ. 2) CPVMX = RESULT(1)
              IF (I .EQ. 4) HVMX = RESULT(1)
              GO TO 20
            END IF
            IF (J .EQ. 2) THEN
              IF (I .EQ. 2) CPLMX = RESULT(1)
              IF (I .EQ. 4) HLMX = RESULT(1)
              GO TO 20
            END IF
          END IF
          DO 30 K = 1, N
            IF (I.EQ.1 .AND. J.EQ.1) CPV(K) = RESULT(K)
            IF (I.EQ.1 .AND. J.EQ.2) CPL(K) = RESULT(K)
            IF (I.EQ.3 .AND. J.EQ.1) HV(K) = RESULT(K)
            IF (I.EQ.3 .AND. J.EQ.2) HL(K) = RESULT(K)
30          CONTINUE
20        CONTINUE
10      CONTINUE

      RETURN
      END

```

4 Calling Utility Subroutines

User subroutines can call Aspen Plus utility subroutines for specific tasks. This chapter describes the following Aspen Plus utilities:

Packing Utilities

Utility	Use
SHS_CPACK	Packs a conventional substream vector before calling a physical property monitor
SHS_NCPACK	Packs a nonconventional substream vector before calling a physical property monitor
SHS_SCPACK	Packs lists of conventional component flow rates from several phases into an output array which contains mole fractions of components that are present in at least one of the phases

Error Handling Utilities

Utility	Use
DMS_IRRCHK and DMS_ERRPRT	Issues error and information messages during user subroutine calculations

Utilities for Writing Messages

Utility	Use
ZREP_RPTHDR	Paginates when writing to the report file
DMS_WRTRM	Writes messages to the Control Panel or Terminal File

Component Identification Utilities

Utility	Use
DMS_KFORM or DMS_KFORMC	Determines the sequence number (IDX) of a conventional component from its alias
DMS_KCCID or DMS_KCCIDC	Determines the sequence number (IDX) of a conventional component from its ID
DMS_KNCID or DMS_KNCIDC	Determines the sequence number (IDX) of a nonconventional component from its ID
PPUTL_IDXCS2	Determines CAS number for a given component alias or vice versa

Component Attribute Information Utilities

Utility	Use
SHS_CAELID	Find a component attribute element ID given the attribute ID and the element number
SHS_CAID	Find a component attribute ID given the component sequence number, the attribute type number, and the substream structure
SHS_LCAOFF	Find the offset of a component attribute from the beginning of the substream given the structure of the substream, the component sequence number and the attribute type number
SHS_LCATT	Find the offset of a component attribute from the substream given the structure of the substream, attributed component sequence number and the attribute ID
SHS_NCAVAR	Find the number of elements in a component attribute given the attribute type index, the component index, and the substream type

Component Attribute Calculation Utilities

Utility	Use
SHS_CAMIX	Mix the attributes from two inlet streams into an outlet stream
SHS_CASPLT	Calculate the attribute values in a product stream produced by splitting a given feed stream
SHS_CASPSS	Calculate the attribute values in a product substream produced by splitting a given feed substream
SHS_CAUPDT	Calculate class zero polymer component attribute values in a product stream based on class 2 component attributes.

Polymer Property Utilities

Utility	Use
POLY_GETCRY	Get the crystallinity of a list of components
POLY_GETDPN	Get the number average degree of polymerization
POLY_GETMWN	Calculate the true molecular weight of a polymer from the degree of polymerization and the average segment molecular weight
POLY_GETMWW	Get the weight-average molecular weight vector for polymer components, or the component molecular weight for standard components
POLY_GETPDI	Get the polydispersity index for polymer components. For standard components, parameter POLPDI is returned.
POLY_GETSMF	Get the segment mole fractions for a polymer or oligomer component
POLY_GETSWF	Get the segment weight (mass) fractions for a polymer or oligomer component

Polymer Type Utilities

Utility	Use
PPUTL_ISCAT	Determine whether a component is a catalyst
PPUTL_ISINI	Determine whether a component is an ionic initiator
SHS_ISOLIG	Determine whether a component is an oligomer

Utility	Use
SHS_ISPOLY	Determine whether a component is a polymer
PPUTL_ISSEG	Determine whether a component is a segment

Polymer Component Fraction Utilities

Utility	Use
POLY_XATOWT	Get true weight fractions for all of the components present
POLY_XATOXT	Get true mole fractions for all of the components present

General Stream Handling Utilities

Utility	Use
SHS_IPTYPE	Get the substream type number from the stream class descriptor bead
SHS_LPHASE	Find the offset of a substream from the beginning of a stream structure
SHS_NPHASE	Find the number of substreams from the stream class descriptor bead
SHS_NSVAR	Find the number of stream variables
SHS_SSCOPY	Copy substream information from one stream to another

Plex Offset Utility

Utility	Use
DMS_IFCMN or DMS_IFCMNC	Determines DMS_PLEX offsets for component data areas

Other Utilities

Utility	Use
UU_GETP_AMB	Obtain the ambient pressure
DMS_IAPCBR	Check whether break key has been pressed
PPUTL_CHKCOMPNAME	Obtain company name from license

Packing Utilities

Packing sets up a mole fraction vector for the components actually present, eliminating zero flow components. SHS_CPACK is a utility routine for packing conventional component flows, given a conventional substream (type MIXED or CISOLID).

SHS_NCPACK is a utility routine for packing nonconventional component flows, given a nonconventional substream (type NC).

SHS_SCPACK packs flows when the vectors for individual phases are known.

Calling Sequence for SHS_CPACK

CALL SHS_CPACK (SUBSTR, NCP, IDX, X, FLOW)

Calling Sequence for SHS_NCPACK

CALL SHS_NCPACK (SUBSTR, NCP, IDX, X, FLOW)

Calling Sequence for SHS_SCPACK

SUBROUTINE SHS_SCPACK (NPHASE, LPHASE, NCP, IDX, XV, FLOWV)

Argument List Descriptions for Packing Utilities

Variable	I/O [†]	Type	Dimension	Description
SUBSTR	I	REAL*8	(1)	Vector of component flows
NPHASE	I	INTEGER	---	Number of phases to be packed
LPHASE	I	INTEGER	NPHASE	Vector of phase plex addresses
NCP	O	INTEGER	—	Number of components actually present
IDX	O	INTEGER	NCP	Component index vector
X	O	REAL*8	NCP	Packed mole fraction vector for SHS_CPACK; mass fraction vector for SHS_NCPACK
XV	O	REAL	NCP, NPHASE	Mole fraction array of components actually present
FLOW	O	REAL*8	—	Total substream flow (kgmole/s for SHS_CPACK; kg/s for SHS_NCPACK)
FLOWV	O	REAL	NPHASE	Vector of phase flow rates

[†]I = Input to subroutine, O = Output from subroutine

Aspen Plus Error Handler

The Aspen Plus Error Handler consists of two subprograms:

- Function DMS_IRRCHK.
- Subroutine DMS_ERRPRT.

User-written subroutines should call DMS_ERRPRT when they find an error to report. Make sure your subroutine performs these steps:

- 1** Call function DMS_IRRCHK to determine if the message should be printed. DMS_IRRCHK determines this based upon the error severity, diagnostic level, and number of errors that have already been printed.
- 2** If DMS_IRRCHK=1, write a message using Fortran internal WRITE statements into the message buffer in COMMON / DMS_ERROUT / (see The Fortran WRITE Statement, this chapter.).
- 3** If DMS_IRRCHK=1, call DMS_ERRPRT to print the message.

Note: This WRITE statement will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

Calling Sequence for Error Handler

```

INTEGER DMS_IRRCHK
.
.
.
IPRT=DMS_IRRCHK (IPROG, ISEV, ICODE, LDIAG, LTERM, IESCAL,
                IPHYS,NHEAD)
CALL DMS_ERRPRT (NLINES)

```

Argument List Descriptions for Error Handler

Variable	I/O [†]	Type	Dimension	Description
IPRT	O	INTEGER	—	Print flag, IPRT=0, don't print message IPRT=1, print message
IPROG	I	INTEGER	2	Calling subroutine name (two integer words, 4 characters in first, two characters in second)
ISEV	I	INTEGER	—	Error severity If ISEV=-1, system error If ISEV=0, terminal error If ISEV=1, severe error If ISEV=2, error If ISEV=3, warning If ISEV≥4, information
ICODE	I	INTEGER	—	Error number; must have fewer than five digits and be unique within calling subroutine
LDIAG	I	INTEGER	—	Diagnostic level
LTERM	I	INTEGER	—	Terminal diagnostic level. Not implemented; set LTERM=USER_IUMISS from COMMON/PPEXEC_USER
IESCAL	I	INTEGER	—	Error escalation level. Not implemented; set IESCAL= 0
IPHYS	I	INTEGER	—	If IPHYS=1, calling subroutine is a physical property routine If IPHYS=0, calling subroutine is not a physical property routine
NHEAD	I	INTEGER	—	Amount of header information printed If NHEAD=0, no header If NHEAD=1, cursory header If NHEAD=2, normal header
NLINES	I	INTEGER	—	Number of lines to print

[†]I = Input to subroutine, O = Output from subroutine

Report Header Utility

The Aspen Plus report file is designed for pages with a fixed number of lines, specified on the **Setup | Report Options | General** sheet. Subroutine ZREP_RPTHDR handles pagination, page header information, and table of contents entries. A user-written subroutine or a Fortran block that will

execute during the report pass should call ZREP_RPTHDR *before* it writes to the Aspen Plus report file.

Calling Sequence for ZREP_RPTHDR

CALL ZREP_RPTHDR (LINES, IPAGE, ISECT, ISUB)

Argument List Descriptions for ZREP_RPTHDR

Variable	I/O [†]	Type	Dimension	Description
LINES	I	INTEGER	—	Number of lines to be printed
IPAGE	I	INTEGER	—	IPAGE=0, ZREP_RPTHDR places output on current page if there is enough room. Otherwise, ZREP_RPTHDR places output on a new page.
ISECT	I	INTEGER	—	Report section number for table of contents and header (See ISECT)
ISUB	I	INTEGER	10	Integer array containing the report subsection heading for the table of contents. ZREP_RPTHDR makes a new table of contents entry whenever it is called with a new ISUB. COMMON /DMS_RPTGLB/ contains an array of the correct length which can be passed to ZREP_RPTHDR (see Appendix A). For certain applications, such as the USER and USER2 unit operation models, Aspen Plus initializes ISUB in COMMON /DMS_RPTGLB/ and the user routine need not change it.

[†]I = Input to subroutine, O = Output from subroutine

ISECT

ISECT can take the following values:

- 1 Flowsheet section
- 2 Physical property section
- 3 Unit operation block section
- 4 Stream section
- 5 Cost block section
- 6 Economic evaluation section
- 7 Data regression section
- 8 Physical property tables section
- 9 Sensitivity block section
- 10 ADA/PCS section
- 11 Utility section
- 12 Input section
- 13 Property constant estimation section
- 14 Balance block section

Terminal File Writer Utility

If you want to write to the Terminal File (when running Aspen Plus from command line without the Windows user interface) or to the Control Panel (when running Aspen Plus from the Windows user interface) follow these steps:

- 1 Include the following (beginning in column 1):

```
#include "dms_maxwrt.cmn"
```
- 2 Use a Fortran internal write to MAXWRT_MAXBUF and then call DMS_WRTTRM, with the number of lines to be written as the argument.

Calling Sequence for Terminal File Writer

```
CALL DMS_WRTTRM (NLINES)
```

Argument List Descriptions for Terminal File Writer

Variable	I/O [†]	Type	Dimension	Description
NLINES	I	INTEGER	—	Number of lines to print

[†]I = Input to subroutine, O = Output from subroutine

Example: User Subroutine Writing to the Terminal

```
SUBROUTINE USR001
.
.
.
IMPLICIT REAL*8 (A-H, O-Z)
#include "dms_maxwrt.cmn"

DIMENSION ID(2)
.
.
.
C
C   NOW WRITE TO THE TERMINAL
C
1000 FORMAT ('FLASH OF STREAM', 2A4, 'FAILED')
WRITE (MAXWRT_MAXBUF, 1000) ID(1), ID(2)
CALL DMS_WRTTRM(1)
.
.
.
RETURN
END
```

Utilities to Determine Component Index

It is sometimes useful to determine the sequence number of a conventional or nonconventional component among all the conventional or nonconventional components you entered on the **Components | Specifications | Selection** sheet. All physical property data arrays, the stream vectors, and certain other component arrays are packed in component list order. The data for all components are stored in contiguous arrays, from 1 to NCOMP_NCC for conventional, and from 1 to NCOMP_NNCC for nonconventional components (see COMMON DMS_NCOMP, Appendix A). Also see Appendix C for a description of the structure of the stream vector.

To locate a specific component in such arrays:

Call	For a	If you know its
DMS_KFORM or DMS_KFORMC	Conventional component	Alias
DMS_KCCID or DMS_KCCIDC	Conventional component	ID
DMS_KNCID or DMS_KNCIDC	Nonconventional component	ID

Use the versions with names ending in C if you specify the alias as a CHARACTER variable.

Calling Sequence for DMS_KFORM

```
INTEGER DMS_KFORM
...
K = DMS_KFORM (NAME)
```

Calling Sequence for DMS_KFORMC

```
INTEGER DMS_KFORMC
...
K = DMS_KFORMC (CNAME)
```

Calling Sequence for DMS_KCCID

```
INTEGER DMS_KCCID
...
K = DMS_KCCID (IDC)
```

Calling Sequence for DMS_KCCIDC

```
INTEGER DMS_KCCIDC
...
K = DMS_KCCIDC (CIDC)
```

Calling Sequence for DMS_KNCID

```
INTEGER DMS_KNCID
...
K = DMS_KNCID (IDNC)
```

Calling Sequence for DMS_KNCIDC

```
INTEGER DMS_KNCIDC
...
K = DMS_KNCIDC (CIDNC)
```

Argument List Descriptions for Component Index Utilities

Variable	I/O [†]	Type	Dimension	Description
NAME	I	INTEGER	3	Component alias as 3 integer words
CNAME	I	CHARACTER*12	—	Component alias as a character string
IDC	I	INTEGER	2	Conventional component ID as two integer words
CIDC	I	CHARACTER*8	—	Conventional component ID as a character string
IDNC	I	INTEGER	2	Nonconventional component ID as two integer words
CIDNC	I	CHARACTER*8	—	Nonconventional component ID as a character string
K	O	INTEGER	—	Component index If = 0, the component has not been declared on the Components Specifications Selection sheet

[†]I = Input to subroutine, O = Output from subroutine

Component Index Example

In a User unit operation model (see Chapter 5), the packed stream vectors for the attached streams are supplied as arguments to the Fortran model. In this example, we are interested in the amount of water in the MIXED substream of the first material inlet stream (SIN1):

```
...
INTEGER DMS_KFORMC
INTEGER IWATER
REAL*8 WATER
IWATER=DMS_KFORMC('H2O')
IF (IWATER.NE.0) THEN
    WATER=SIN1(IWATER)
ELSE
    WATER=0.0
ENDIF
...
```

CAS Number Utility

If you know only the CAS registry number for a component and you want its component alias, or you know only the alias and want CAS number, use this utility to find the other identifier.

Calling Sequence for PPUTL_IDXCS2

CALL PPUTL_IDXCS2 (IALIAS, ICAS, ITYPE, IOK)

Argument List Descriptions for CAS Number Utility

Variable	I/O [†]	Type	Dimension	Description
IALIAS	I/O	INTEGER	3	Component alias as 3 integer words
ICAS	I/O	INTEGER	4	CAS number (string encoded as 4 integer words)
ITYPE	I	INTEGER	-	If 1, find ICAS for given IALIAS. If 2, find IALIAS for given ICAS.
IOK	O	INTEGER	-	If 0, no match was found. If 1, match was found and value returned.

[†]I = Input to subroutine, O = Output from subroutine

Component Attribute Information Utilities

These utilities are useful for handling stream data with component attributes, as commonly encountered with nonconventional components and polymers.

Utility	Use
SHS_CAELID	Find a component attribute element ID given the attribute ID and the element number
SHS_CAID	Find a component attribute ID given the component sequence number, the attribute type number, and the substream structure
SHS_LCAOFF	Find the offset of a component attribute from the beginning of the substream given the structure of the substream, the component sequence number and the attribute type number
SHS_LCATT	Find the offset of a component attribute from the substream given the structure of the substream, attributed component sequence number and the attribute ID
SHS_NCAVAR	Find the number of elements in a component attribute given the attribute type index, the component index, and the substream type

Calling Sequence for SHS_CAELID

SUBROUTINE SHS_CAELID (IDCAT, IELEM, IDCAEL)

Argument List Descriptions for SHS_CAELID

Variable	I/O	Type	Dimension	Description
IDCAT	I	INTEGER	2	Comp attr. ID in two integer words
IELEM	I	INTEGER	---	Comp attr. element no.
IDCAEL	O	INTEGER	2	Comp attr. element ID in two integer words

Example of Calling SHS_CAELID in User Routine

```
INTEGER IDCAT(2), IELEM, IDCAEL(2)
DATA IDCAT / 'DPN ', ' ' /
IELEM=1
.
.
.
CALL SHS_CAELID ( IDCAT, IELEM, IDCAEL )
```

Calling Sequence for SHS_CAID

```
SUBROUTINE SHS_CAID ( ISSCNC, NCSEQ, J, IDCAT )
```

Argument List Descriptions for SHS_CAID

Variable	I/O	Type	Dimension	Description
ISSCNC	I	INTEGER	---	Flag: 1 = conventional substream 2 = nonconventional substream
NCSEQ	I	INTEGER	---	Attributed component sequence number
J	I	INTEGER	---	Comp attr. type no.
IDCAT	O	INTEGER	2	Comp attr. ID in two integer words

Example of Calling SHS_CAID in User Routine

```
INTEGER ISSCNC, NCSEQ, J, IDCAT(2)
ISSCNC=1                !'Conventional substream'
J=1
.
.
.
CALL SHS_CAID ( ISSCNC, NCSEQ, J, IDCAT )
```

Calling Sequence for SHS_LCAOFF

```
FUNCTION SHS_LCAOFF (ISSCNC, NCSEQ, J)
```

Argument List Descriptions for SHS_LCAOFF

Variable	I/O	Type	Dimension	Description
ISSCNC	I	INTEGER	---	Flag: 1 = Conventional substream 2 = non conventional substream
NCSEQ	I	INTEGER	---	Attributed component sequence number
J	I	INTEGER	---	Comp. attribute type no.
LCAOFF	O	INTEGER	---	Attribute offset from substream

Example of Calling SHS_LCAOFF in User Routine

```
INTEGER ISSCNC, NCSEQ, J, SHS_LCAOFF
ISSCNC=1                !'Conventional substream'
NCSEQ=2                 !'Second attributed component'
```

```

J=1          !'First component attribute in
              the list for this component'
.
.
.
N = SHS_LCAOFF (ISSCNC, NCSEQ, J)

```

Calling Sequence for SHS_LCATT

```
FUNCTION SHS_LCATT (ISSCNC, NCSEQ, IDCATT)
```

Argument List Descriptions for SHS_LCATT

Variable	I/O	Type	Dimension	Description
ISSCNC	I	INTEGER	---	Flag: 1 = conventional substream 2 = non conventional substream
NCSEQ	I	INTEGER	---	Attributed component sequence number
IDCATT	I	INTEGER	2	Component attribute ID in two integer words
LCATT	O	INTEGER	---	Attribute offset

Example of Calling SHS_LCATT in User Routine

```

INTEGER ISSCNC, NCSEQ, IAID(2), SHS_LCATT
DATA IAD / 'DPN ', / ' ' /
ISSCNC=1
NCSEQ=1
.
.
.
N = SHS_LCATT (ISSCNC, NCSEQ, IAID)

```

Calling Sequence for SHS_NCAVAR

```
FUNCTION SHS_NCAVAR (ISSCNC, NCSEQ, J)
```

Argument List Descriptions for SHS_NCAVAR

Variable	I/O	Type	Dimension	Description
ISSCNC	I	INTEGER	---	Flag: 1 = conventional substream 2 = non conventional substream
NCSEQ	I	INTEGER	---	Attributed component sequence number
J	I	INTEGER	---	Comp attribute type no.
NCAVAR	O	INTEGER	---	Attribute length

Example of Calling SHS_NCAVAR in User Routine

```

INTEGER ISSCNC, NCSEQ, J, SHS_NCAVAR
ISSCNC=1          !'Conventional substream'
NCSEQ=1

```

```

J=1
.
.
.
N = SHS_NCAVAR ( ISSCNC, NCSEQ, J)

```

Component Attribute Calculation Utilities

These utilities perform the calculations Aspen Plus would perform when handling component attributes for nonconventional and polymer components.

Utility	Use
SHS_CAMIX	Mix the attributes from two inlet streams into an outlet stream
SHS_CASPLT	Calculate the attribute values in a product stream produced by splitting a given feed stream
SHS_CASPSS	Calculate the attribute values in a product substream produced by splitting a given feed substream
SHS_CAUPDT	Calculate class zero polymer component attribute values in a product stream based on class 2 component attributes.

Calling Sequence for SHS_CAMIX

```
SUBROUTINE SHS_CAMIX ( IP, SS1, SS2, SSO )
```

Argument List Descriptions for SHS_CAMIX

Variable	I/O	Type	Dimension	Description
IP	I	INTEGER	---	Substream type 1 or 2 = conventional 3 = nonconventional
SS1	I	REAL*8	1	1st input substream
SS2	I	REAL*8	1	2nd input substream
SSO	O	REAL*8	1	Output substream

Example of Calling SHS_CAMIX in User Routine

```

INTEGER IP
REAL*8 SS1(1), SS2(1), SS0(1)
IP=1          !'Conventional substream'
.
.
.
CALL SHS_CAMIX ( IP, SS1, SS2, SSO )

```

Calling Sequence for SHS_CASPLT

```
SUBROUTINE SHS_CASPLT ( FEED, PROD, NSUBS, IDXSUB, ITYPE )
```

Argument List Descriptions for SHS_CASPLT

Variable	I/O	Type	Dimension	Description
FEED	I	REAL*8	(1)	Combined feed stream
PROD	O	REAL*8	(1)	Given outlet stream
NSUBS	I	INTEGER	---	Number of substreams
ITYPE	I	INTEGER	NSUB	Substream type vector
IDXSUB	I	INTEGER	NSUB	Substream index vector

Example of Calling SHS_CASPLT in User Routine

```
INTEGER NSUBS, ITYPE(1), IDXSUB(1)
REAL*8 FEED(1), PROD(1)
IDXSUB(1)=1                !'First substream'
ITYPE(1)=1                 !'Conventional'
.
.
.
CALL SHS_CASPLT ( FEED, PROD, NSUBS, IDXSUB, ITYPE )
```

Calling Sequence for SHS_CASPSS

```
SUBROUTINE SHS_CASPSS ( FEED, PROD, ITYPE )
```

Argument List Descriptions for SHS_CASPSS

Variable	I/O	Type	Dimension	Description
FEED	I	REAL*8	(1)	Feed substream
PROD	O	REAL*8	(1)	Outlet substream
ITYPE	I	INTEGER	NSUB	Substream type

Example of Calling SHS_CASPSS in User Routine

```
INTEGER ITYPE(1)
REAL*8 FEED(1), PROD(1)
ITYPE(1)=1                !'Conventional'
.
.
.
CALL SHS_CASPSS ( FEED, PROD, ITYPE )
```

Calling Sequence for SHS_CAUPDT

```
SUBROUTINE SHS_CAUPDT ( STREAM, NSUBS, IDXSUB, IPHASE )
```

Argument List Descriptions for SHS_CAUPDT

Variable	I/O	Type	Dimension	Description
STREAM	I	REAL*8	(1)	Stream vector
NSUBS	I	INTEGER	---	Number of substreams
IDXSUB	I	INTEGER	NSUBS	Substream index vector

Variable	I/O	Type	Dimension	Description
IPHASE	I	INTEGER	NSUBS	Substream type vector 1= MIXED 2= CISOLID 3= NCSOLID

Example of Calling SHS_CAUPDT in User Routine

```

INTEGER NSUBS, IDXSUB(1), IPHASE(1)
REAL*8 STREAM(1)
IPHASE(1)=1           !'First phase'
.
.
.
CALL SHS_CAUPDT ( STREAM, NSUBS, IDXSUB, IPHASE )

```

Polymer Property Utilities

These utilities calculate specific polymer properties.

Polymer Property Utilities

Utility	Use
POLY_GETCRY	Get the crystallinity of a list of components
POLY_GETDPN	Get the number average degree of polymerization
POLY_GETMWN	Calculate the true molecular weight of a polymer from the degree of polymerization and the average segment molecular weight
POLY_GETMWW	Get the weight-average molecular weight vector for polymer components, or the component molecular weight for standard components
POLY_GETPDI	Get the polydispersity index for polymer components. For standard components, parameter POLPDI is returned.
POLY_GETSMF	Get the segment mole fractions for a polymer or oligomer component
POLY_GETSWF	Get the segment weight (mass) fractions for a polymer or oligomer component

Calling Sequence for POLY_GETCRY

```
SUBROUTINE POLY_GETCRY ( NCNC, NCP, IDX, CRY )
```

Argument List Descriptions for POLY_GETCRY

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
NCP	I	INTEGER	---	Number of components
IDX	I	INTEGER	NCP	Component index vector

Variable	I/O	Type	Dimension	Description
CRY	O	INTEGER	NCP	Crystalline fraction

Example of Calling POLY_GETCRY in User Routine

```

INTEGER NCNC, NCP, IDXP, CRY(1)
NCNC=1                !'Conventional substream'
NCP=1
IDXP=2                !'Polymer is 2nd component'
.
.
.
CALL POLY_GETCRY( NCNC, NCP, IDX, CRY )

```

Calling Sequence for POLY_GETDPN

```
SUBROUTINE POLY_GETDPN ( NCNC, NCP, IDX, DPN )
```

Argument List Descriptions for POLY_GETDPN

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
NCP	I	INTEGER	---	Number of components
IDX	I	INTEGER	NCP	Component index vector
DPN	O	REAL*8	NCP	Degree of polymerization

Example of Calling POLY_GETDPN in User Routine

```

INTEGER NCNC, NCP, IDXP
REAL*8 DPN
NCNC=1                !'Conventional substream'
NCP=1
IDXP=2                !'Polymer is 2nd component'
.
.
.
CALL POLY_GETDPN( NCNC, NCP, IDX, DPN )

```

Calling Sequence for POLY_GETMWN

```
SUBROUTINE POLY_GETMWN ( NCNC, NCP, IDX, XMWTRU )
```

Argument List Descriptions for POLY_GETMWN

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
NCP	I	INTEGER	---	Number of components
IDX	I	INTEGER	NCP	Component index vector
XMWTRU	O	REAL*8	NCP	True number average molecular weight

Example of Calling POLY_GETMWN in User Routine

```
INTEGER NCNC, NCP, IDXP
REAL*8 XMWTRU(1)
NCNC=1                !'Conventional substream'
NCP=1
IDXP=2                !'Polymer is 2nd component'
.
.
.
CALL POLY_GETMWN( NCNC, NCP, IDX, XMWTRU )
```

Calling Sequence for POLY_GETMWW

```
SUBROUTINE POLY_GETMWW ( NCNC, NCP, IDX, MWW )
```

Argument List Descriptions for POLY_GETMWW

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
NCP	I	INTEGER	---	Number of components present
IDX	I	INTEGER	NCP	Component index vector
MWW	O	REAL*8	NCP	Weight-average molecular weight

Example of Calling POLY_GETMWW in User Routine

```
REAL*8 MWW
INTEGER NCNC, NCP, IDX
NCNC=1                !'Conventional substream'
NCP=1                  !'Only one component (polymer)'
IDXI=2                !'Polymer is 2nd component'
.
.
.
CALL POLY_GETMWW( NCNC, NCP, IDX, MWW )
```

Calling Sequence for POLY_GETPDI

```
SUBROUTINE POLY_GETPDI ( NCNC, NCP, IDX, MWW )
```

Argument List Descriptions for POLY_GETPDI

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
NCP	I	INTEGER	---	Number of components present
IDX	I	INTEGER	NCP	Component index vector
PDI	O	REAL*8	NCP	Polydispersity index

Example of Calling POLY_GETPDI in User Routine

```
INTEGER NCNC, NCP, IDXI
REAL*8 PDI
NCNC=1          !'Conventional substream'
NCP=1           !'Only one component (polymer)'
IDXI=2          !'Polymer is 2nd component'
.
.
.
CALL POLY_GETPDI( NCNC, NCP, IDXI, PDI )
```

Calling Sequence for POLY_GETSMF

```
SUBROUTINE POLY_GETSMF ( NCNC, IDX, SMFRAC )
```

Argument List Descriptions for POLY_GETSMF

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
IDX	I	INTEGER	---	Component index
SMFRAC	O	REAL*8	NCOMP_NSEG	Segment mole fractions

The number of segments is retrieved from common DMS_NCOMP. The SMFRAC variable must be dimensioned to NCOMP_NSEG or larger.

Example of Calling POLY_GETSMF in User Routine

```
INTEGER NCNC, NCP, IDXI
REAL*8 SMFRAC(10) ! dimension must be > Nseg
NCNC=1          !'Conventional substream'
IDXI=2          !'Polymer is 2nd component'
.
.
.
CALL POLY_GETSMF( NCNC, IDXI, SMFRAC)
```

Calling Sequence for POLY_GETSWF

```
SUBROUTINE POLY_GETSWF ( NCNC, IDX, SWFRAC )
```

Argument List Descriptions for POLY_GETSWF

Variable	I/O	Type	Dimension	Description
NCNC	I	INTEGER	---	1 = conventional substream 2 = non-conventional substream
IDX	I	INTEGER	---	Component index
SWFRAC	O	REAL*8	NCOMP_NSEG	Segment mole fractions

The number of segments is retrieved from common DMS_NCOMP. The SWFRAC variable must be dimensioned to NCOMP_NSEG or larger.

Example of Calling POLY_GETSWF in User Routine

```
INTEGER NCNC, NCP, IDXI
REAL*8 SWFRAC(10) ! dimension must be > Nseg
NCNC=1             !'Conventional substream'
IDXI=2             !'Polymer is 2nd component'
.
.
.
CALL POLY_GETSWF( NCNC, IDXI, SWFRAC)
```

Polymer Type Utilities

These utilities determine the type of a component for polymer component types.

Utility	Use
PPUTL_ISCAT	Determine whether a component is a catalyst
PPUTL_ISINI	Determine whether a component is an ionic initiator
SHS_ISOLIG	Determine whether a component is an oligomer
SHS_ISPOLY	Determine whether a component is a polymer
PPUTL_ISSEG	Determine whether a component is a segment

Calling Sequence for PPUTL_ISCAT

```
FUNCTION PPUTL_ISCAT ( ICOMP )
```

Argument List Descriptions for PPUTL_ISCAT

Variable	I/O	Type	Dimension	Description
ICOMP	I	INTEGER	---	Component index
ISCAT	O	LOGICAL	---	True for catalysts

Example of Calling PPUTL_ISCAT in User Routine

```
INTEGER ICOMP
LOGICAL PPUTL_ISCAT
ICOMP=2      !
.
.
.
If ( PPUTL_ISCAT( ICOMP ) ) then...
```

Calling Sequence for PPUTL_ISINI

```
FUNCTION PPUTL_ISINI ( ICOMP )
```

Argument List Descriptions for PPUTL_ISINI

Variable	I/O	Type	Dimension	Description
ICOMP	I	INTEGER	---	Component index

Variable	I/O	Type	Dimension	Description
ISINI	O	LOGICAL	---	True for initiators

Example of Calling PPUTL_ISINI in User Routine

```

INTEGER ICOMP
LOGICAL PPUTL_ISINI
ICOMP=2      !
.
.
.
If ( PPUTL_ISINI( ICOMP ) ) then...
```

Calling Sequence for SHS_ISOLIG

```
FUNCTION SHS_ISOLIG ( ICOMP )
```

Argument List Descriptions for SHS_ISOLIG

Variable	I/O	Type	Dimension	Description
ICOMP	I	INTEGER	---	Component index
ISOLIG	O	LOGICAL	---	True for oligomers

Example of Calling SHS_ISOLIG in User Routine

```

INTEGER ICOMP
LOGICAL SHS_ISOLIG
ICOMP=3      !
.
.
.
If ( SHS_ISOLIG( ICOMP ) ) then...
```

Calling Sequence for SHS_ISPOLY

```
FUNCTION SHS_ISPOLY ( ICOMP )
```

Argument List Descriptions for SHS_ISPOLY

Variable	I/O	Type	Dimension	Description
ICOMP	I	INTEGER	---	Component index
ISPOLY	O	LOGICAL	---	True for polymers

Example of Calling SHS_ISPOLY in User Routine

```

INTEGER ICOMP
LOGICAL SHS_ISPOLY
ICOMP=2      !
.
.
.
If ( ISPOLY( ICOMP ) ) then...
```

Calling Sequence for PPUTL_ISSEG

```
FUNCTION PPUTL_ISSEG ( ICOMP )
```

Argument List Descriptions for PPUTL_ISSEG

Variable	I/O	Type	Dimension	Description
ICOMP	I	INTEGER	---	Component index
ISSEG	O	LOGICAL	---	True for segments

Example of Calling PPUTL_ISSEG in User Routine

```
INTEGER ICOMP
LOGICAL PPUTL_ISSEG
ICOMP=5      !
.
.
.
If ( PPUTL_ISSEG( ICOMP ) ) then...
```

Polymer Component Fraction Utilities

Use these routines to convert apparent mole fraction vectors to other types of fractions.

Utility	Use
POLY_XATOWT	Get true weight fractions for all of the components present
POLY_XATOXT	Get true mole fractions for all of the components present

Calling Sequence for POLY_XATOWT

```
SUBROUTINE POLY_XATOWT ( X, NCP, IDX, WT )
```

Argument List Descriptions for POLY_XATOWT

Variable	I/O	Type	Dimension	Description
X	I	REAL*8	NCP	Apparent mole fraction vector
NCP	I	INTEGER	---	Number of components present
IDX	I	INTEGER	NCP	Component index vector
WT	O	REAL*8	NCP	True weight fraction vector

Example of Calling POLY_XATOWT in User Routine

```
C    Dimension of WT can be set high enough
C    to avoid overwriting;
C    Alternatively, the proper work space can
C    be assigned to the
C    array WT (preferred method)
C
```

```

REAL*8 WT(10)
.
.
.
CALL POLY_XATOWT (X, NCP, IDX, WT)

```

Calling Sequence for POLY_XATOXT

```
SUBROUTINE POLY_XATOXT ( NCP, IDX, XMW, X, XTRUE )
```

Argument List Descriptions for POLY_XATOXT

Variable	I/O	Type	Dimension	Description
NCP	I	INTEGER	---	Number of components present
IDX	I	INTEGER	NCP	Component index vector
XMW	I	REAL*8	NCC	Molecular weight vector
NCC	I	INTEGER	---	Number of components listed
X	I	REAL*8	NCP	Apparent mole fraction vector
XTRUE	O	REAL*8	NCP	True mole fraction vector

Example of Calling POLY_XATOXT in User Routine

```

C      Dimension of XTRUE can be set high
C      enough to avoid overwriting;
C      Alternatively, the proper work space
C      can be assigned to the
C      array XTRUE (preferred method)
C
      REAL*8 XTRUE(10)
      .
      .
      .
      CALL POLY_XATOXT (NCP, IDX, XMW, X, XTRUE)

```

General Stream Handling Utilities

These utilities operate on stream class descriptor beads available in models such as User2.

Utility	Use
SHS_IPTYPE	Get the substream type number from the stream class descriptor bead
SHS_LOCPD	Locate PSD information in a substream and locate size intervals
SHS_LPHASE	Find the offset of a substream from the beginning of a stream structure
SHS_NPHASE	Find the number of substreams from the stream class descriptor bead
SHS_NSVAR	Find the number of stream variables

Utility	Use
SHS_SSCOPY	Copy substream information from one stream to another

Calling Sequence for SHS_IPTYPE

```
FUNCTION SHS_IPTYPE (LD, I)
```

Argument List Descriptions for SHS_IPTYPE

Variable	I/O	Type	Dimension	Description
LD	I	INTEGER	---	Address of stream class descriptor bead
I	I	INTEGER	---	Substream number

Example of Calling SHS_IPTYPE in User Routine

```
INTEGER I, SHS_IPTYPE
INTEGER LD      !'Available in user models such as USER2'
I=1             !'Substream 1'
.
.
.
N = SHS_IPTYPE (LD, I)
```

Calling Sequence for SHS_LOCPSD

```
SUBROUTINE SHS_LOCPSD(LD, J, LPSD, NSI, LSLIM)
```

Argument List Descriptions for SHS_LOCPSD

Variable	I/O	Type	Dimension	Description
LD	I	INTEGER	---	Address of stream class descriptor bead
J	I	INTEGER	---	Substream index number (See Stream Structure)
LPSD	O	INTEGER	---	Offset of substream I in the stream structure
NSI	O	INTEGER	---	Number of size intervals
LSLIM	O	INTEGER	---	Location of size intervals in the plex

Example of Calling SHS_LOCPSD in User Routine

```
INTEGER I
INTEGER LD      !'Available in user models such as USER2'
REAL*8 SIN1     !'Material stream, as in USER'
#include "dms_plex.cmn"
    equivalence(ib(1),b(1))
#include "ppexec_user.cmn"
C See Appendix A for information about the common blocks
C included above; ppexec_user is included here only to access
C the report file unit number, but dms_plex and the equivalence
C statement are required to access the size intervals in the
C plex.
```

```

.
.
.
      call SHS_LOCPD(LD, 2, LPSD, NSI, LSLIM)
      do i=1, nsi+1
        write(USER_NRPT,*) 'i, slim(i) ', i, b(lslim+i-1)
      enddo
      do i=1, nsi
        write(USER_NRPT,*) 'i, frac(i) ', i, sin1(lpsd+i-1)
      enddo

```

Calling Sequence for SHS_LPHASE

```
FUNCTION SHS_LPHASE (LD, I)
```

Argument List Descriptions for SHS_LPHASE

Variable	I/O	Type	Dimension	Description
LD	I	INTEGER	---	Address of stream class descriptor bead
LPHASE	O	INTEGER	---	Offset of substream I in the stream structure
I	I	INTEGER	---	Substream number

Example of Calling SHS_LPHASE in User Routine

```

INTEGER I, SHS_LPHASE
INTEGER LD      !'Available in user models such as USER2'
I=1
.
.
.
N = SHS_LPHASE (LD, I)

```

Calling Sequence for SHS_NPHASE

```
FUNCTION SHS_NPHASE (LD)
```

Argument List Descriptions for SHS_NPHASE

Variable	I/O	Type	Dimension	Description
LD	I	INTEGER	---	Address of stream
NPHASE	O	INTEGER	---	No. of substreams

Example of Calling SHS_NPHASE in User Routine

```

INTEGER SHS_NPHASE
INTEGER LD      !'Available in user models such as USER2'
.
.
.
N = SHS_NPHASE (LD)

```

Calling Sequence for SHS_NSVAR

```
FUNCTION SHS_NSVAR (LD)
```

Argument List Descriptions for SHS_NSVAR

Variable	I/O	Type	Dimension	Description
LD	I	INTEGER	---	Address of stream class descriptor bead
NSVAR	O	INTEGER	---	Length of stream variable

Example of Calling SHS_NSVAR in User Routine

```
INTEGER SHS_NSVAR
INTEGER LD    !'Available in user models such as USER2'
.
.
.
N = SHS_NSVAR (LD)
```

Calling Sequence for SHS_SSCOPY

```
SUBROUTINE SHS_SSCOPY( LD, S1, S2, IDX, I)
```

Argument List Descriptions for SHS_SSCOPY

Variable	I/O	Type	Dimension	Description
LD	I	INTEGER	---	Locator of descriptor bead
S1	I	REAL*8	(1)	Stream vector to copy from
S2	O	REAL*8	(1)	Stream vector to copy into
IDX	I	INTEGER	---	Location of substream within stream vector
I	I	INTEGER	---	Substream number

Example of Calling SHS_SSCOPY in User Routine

```
INTEGER LD    !'Available in user models such as USER2'
INTEGER IDX(1), I
REAL*8 S(1), S2(1)
I=1          !'Substream 1'
.
.
.
CALL SHS_SSCOPY( LD, S1, S2, IDX, I)
```

Plex Offset Utility

The integer functions DMS_IFCMN and DMS_IFCMNC determine the offset for component data areas in the labeled common DMS_PLEX. This common contains the Plex, the main memory area where all the data for a simulation is stored in a flexible way. The Plex contains data areas sized to hold the

amount of data in the simulation. Each data area has a specific name and length. For example, for molecular weights the name is MW and the length is NCOMP_NCC. (See Appendix A for a list of data areas.)

Calling Sequence for DMS_IFCMN

```
INTEGER DMS_IFCMN
.
.
.
IOFF = DMS_IFCMN (NAME)
```

Calling Sequence for DMS_IFCMNC

```
INTEGER DMS_IFCMNC
CHARACTER*8 CNAME
.
.
.
IOFF = DMS_IFCMNC (CNAME)
```

Argument List Descriptions for DMS_PLEX Offset Utility

Variable	I/O [†]	Type	Dimension	Description
NAME	I	INTEGER	2	Component data name as 2 integer words
CNAME	I	CHARACTER*8	—	Component data name as a character string
IOFF	O	INTEGER	—	Integer or real DMS_PLEX offset. Real data starts at B(IOFF+1). Integer data starts at IB(IOFF+1).

[†]I = Input to subroutine, O = Output from subroutine

Ambient Pressure Utility

When a gauge pressure is used, it is sometimes useful to know the ambient pressure. The ambient pressure can be specified on Setup Specifications Global sheet. UU_GETP_AMB is a utility routine to obtain the ambient pressure in SI units.

Calling Sequence for UU_GETP_AMB

```
CALL UU_GETP_AMB (PAMB)
```

Argument List Descriptions for UU_GETP_AMB Utility

Variable	I/O [†]	Type	Dimension	Description
PAMB	O	REAL	-	Ambient pressure

[†]I = Input to subroutine, O = Output from subroutine

Break Utility

If you write a long, complex model, you may want the ability to stop the model by hitting the break key. Put the following code in your model at each point where you want it to check whether break has been pressed:

Calling Sequence for DMS_IAPCBR

```
INTEGER DMS_IAPCBR, IDUM, IRES
IDUM = 0
IRES = DMS_IAPCBR(IDUM)
```

If IRES comes back with value 0, the break key has NOT been hit. If the value is 1, the break key HAS been hit.

License Name Utility

Use this function to obtain the company name string from the AspenTech license in use. You can use this utility to create and distribute user models which are locked to a particular company.

Calling Sequence for PPUTL_CHKCOMPNAME

```
CALL PPUTL_CHKCOMPNAME (IOK, IDCOMP)
```

Argument List Descriptions for PPUTL_CHKCOMPNAME Utility

Variable	I/O [†]	Type	Dimension	Description
IOK	O	INTEGER	-	Status. 1 on success.
IDCOMP	O	INTEGER	9	Elements 1-8 are the company name. Element 9 is an encrypted checksum and should be ignored.

[†]I = Input to subroutine, O = Output from subroutine

Example of Calling PPUTL_CHKCOMPNAME in User Routine

```
INTEGER IDCOMP(9), IOK
.
.
.
CALL PPUTL_CHKCOMPNAME(IOK, IDCOMP)
WRITE(NNN,999) (IDCOMP(II),II=1,8)
999 FORMAT(6X, 'THIS IS LICENSED COMPANY NAME: ', 8A4)
```

The Fortran WRITE Statement

The Fortran internal WRITE statement is the same as other WRITE statements except that in the internal WRITE statement the integer variable containing

the unit number is replaced by a variable which has been declared a CHARACTER*80 array. Each array element contains space for 80 characters and is considered one record. (Thus, error messages must have fewer than 80 characters per line.) You must give an array name—not an element of an array—or continuation lines will not be accepted. For example, suppose you enter WRITE(IERW(1), 10). A slash in the FORMAT will cause an error, because it would cause continuation past the end of the array element. If you enter WRITE(IERW, 10), no error occurs until the entire array is filled. This is why a separate array is equivalenced to each row in the error message common, COMMON /DMS_ERROUT/ (see Appendix A).

To write an error message with several WRITE statements (one for lines one and two, and another for lines three through five):

This WRITE statement	Should write to
First	IERW1
Second	IERW3

Note: This WRITE statement will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

When you replace the unit number (integer variable) with a character array name, Aspen Plus treats each element in the array as a single output record.

Example: User Subroutine Writing an Error Message

```

SUBROUTINE USR001
.
.
.
IMPLICIT REAL * 8 (A-H,O-Z)
C
#include "dms_errout.cmn"
#include "ppexec_user.cmn"
.
.
.
INTEGER DMS_IRRCHK
DIMENSION IPROG(2)
DATA IPROG /4HUSR0, 4H01 /
CHARACTER*80 IERW1(10), IERW2(9)
CHARACTER*80 IERW3(8), IERW4(7), IERW5(6), IERW6(5)
CHARACTER*80 IERW7(4), IERW8(3), IERW9(2), IERW10
EQUIVALENCE (ERROUT_IEROUT(1),IERW1),(ERROUT_IEROUT(2)
+      ,IERW2), (ERROUT_IEROUT(3),IERW3),(ERROUT_IEROUT(4)
+      ,IERW4), (ERROUT_IEROUT(5),IERW5),(ERROUT_IEROUT(6)
+      ,IERW6), (ERROUT_IEROUT(7),IERW7),(ERROUT_IEROUT(8)
+      ,IERW8), (ERROUT_IEROUT(9),IERW9),(ERROUT_IEROUT(10)
+      ,IERW10)
.
.
.
C
C      ERROR ENCOUNTERED
C
      IF (DMS_IRRCHK(IPROG, 1, 1, KDIAG, USER_IUMISS, 0, 0, 2)
1      .NE. 0) THEN

```

```

C
C          WRITE FIRST TWO LINES OF MESSAGE
C
C          WRITE(IERW1, 10) TEMP, PRES
C
C          WRITE THIRD LINE
C
C          WRITE(IERW3, 20)
C
C          NOW PRINT ALL THREE LINES
C
C          CALL DMS_ERRPRT(3)
C          ENDIF
C          .
C          .
C          .
10  FORMAT(6X, 'CALCULATIONS NOT CONVERGED' /
1    6X, 'TEMP = ', G12.5, 1X, 'PRES = ', G12.5, ' .')
20  FORMAT(6X, 'BLOCK BYPASSED. ')
C          .
C          .
C          .
C          RETURN
C          END

```


5 User Unit Operation Models

This chapter describes the methods for creating custom unit operation models in Aspen Plus:

- Fortran models with User and User2.
- Excel models accessed through User2.
- Fortran models with User3.

User and User2 Fortran Models

The unit operation models User and User2 allow you to interface your own unit operation model with Aspen Plus by supplying a subroutine and entering its name in the Model or Report field on the User or User2 Input Specifications sheet. To use your own equation-oriented models, see User3 Fortran Models on page 100.

The only differences in the argument lists for User and User2 are:

- User can have up to four inlet and four outlet material streams, one information inlet stream, and one information outlet stream.
- User2 has no limit on the number of inlet or outlet streams.

Calling Sequence for User

```
SUBROUTINE subrname†      (NSIN, NINFI, SIN1, SIN2, SIN3, SIN4,  
                           SINFI, NSOUT, NINFO, SOUT1, SOUT2,  
                           SOUT3, SOUT4, SINFO, NSUBS, IDXSUB,  
                           ITYPE, NINT, INT, NREAL, REAL, IDS,  
                           NPO, NBOPST, NIWORK, IWORK, NWORK,  
                           WORK, NSIZE, SIZE, INTSIZ, LD)
```

[†]Subroutine name you entered on the User Input Specifications sheet.

Argument List Descriptions for User

Variable	I/O [†]	Type	Dimension	Description
NSIN	I	INTEGER	—	Number of inlet streams (both material and information)
NINFI	I	INTEGER	—	Number of inlet information streams
SIN1	I/O	REAL*8	(1)	First inlet material stream (see Stream Structure and Calculation Sequence)
SIN2	I/O	REAL*8	(1)	Second inlet material stream (see Stream Structure and Calculation Sequence)
SIN3	I/O	REAL*8	(1)	Third inlet material stream (see Stream Structure and Calculation Sequence)
SIN4	I/O	REAL*8	(1)	Fourth inlet material stream (see Stream Structure and Calculation Sequence)
SINFI	I/O	REAL*8	—	Inlet information stream (see Stream Structure and Calculation Sequence)
NSOUT	I	INTEGER	—	Number of outlet streams (both material and information)
NINFO	I	INTEGER	—	Number of outlet information streams
SOUT1	O	REAL*8	(1)	First outlet material stream (see Stream Structure and Calculation Sequence)
SOUT2	O	REAL*8	(1)	Second outlet material stream (see Stream Structure and Calculation Sequence)
SOUT3	O	REAL*8	(1)	Third outlet material stream (see Stream Structure and Calculation Sequence)
SOUT4	O	REAL*8	(1)	Fourth outlet material stream (see Stream Structure and Calculation Sequence)
SINFO	O	REAL*8	—	Outlet information stream (see Stream Structure and Calculation Sequence)
NSUBS	I	INTEGER	—	Number of substreams in material streams
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block IDs: (* ,1) - Block ID (* ,2) - User model subroutine name (* ,3) - User report subroutine name (* ,4) to (* ,8) - inlet stream IDs (* ,9) to (* ,13) - outlet stream IDs
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)

Variable	I/O [†]	Type	Dimension	Description
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector (see Local Work Arrays)
NSIZE	I	INTEGER	—	Length of size results vector
SIZE	O	REAL*8	NSIZE	Real sizing results (see Size)
INTSIZ	O	INTEGER	NSIZE	Integer size parameters (see Size)
LD	I	INTEGER	—	Plex location of the material stream class descriptor bead

[†]I = Input to subroutine, O = Output from subroutine, W = Workspace

Calling Sequence for User2

```
SUBROUTINE subrname†      (NMATI, SIN, NINFI, SINFI, NMATO, SOUT,
                             NINFO, SINFO, IDSMI, IDSII, IDSMO, IDSIO,
                             NTOT, NSUBS, IDXSUB, ITYPE, NINT, INT,
                             NREAL, REAL, IDS, NPO, NBOPST, NIWORK,
                             IWORK, NWORK, WORK, NSIZE, SIZE, INTSIZ,
                             LD)
```

[†]Subroutine name you entered on the User2 Input Specifications sheet.

Argument List Descriptions for User2

Variable	I/O [†]	Type	Dimension	Description
NMATI	I	INTEGER	—	Number of inlet material streams
SIN	I/O	REAL*8	NTOT, NMATI	Array of inlet material streams (see Stream Structure and Calculation Sequence)
NINFI	I	INTEGER	—	Number of inlet information streams
SINFI	I/O	REAL*8	NINFI	Vector of inlet information streams (see Stream Structure and Calculation Sequence)
NMATO	I	INTEGER	—	Number of outlet material streams
SOUT	O	REAL*8	NTOT, NMATO	Array of outlet material streams
NINFO	I	INTEGER	—	Number of outlet information streams
SINFO	O	REAL*8	NINFO	Vector of outlet information streams (see Stream Structure and Calculation Sequence)
IDSMI	I	INTEGER	2, NMATI	IDs of inlet material streams
IDSII	I	INTEGER	2, NINFI	IDs of inlet information streams
IDSMO	I	INTEGER	2, NMATO	IDs of outlet material streams
IDSIO	I	INTEGER	2, NINFO	IDs of outlet information streams
NTOT	I	INTEGER	—	Length of material streams
NSUBS	I	INTEGER	—	Number of substreams in material streams
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)

Variable	I/O [†]	Type	Dimension	Description
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 3	Block IDs: (* , 1) - Block ID (* , 2) - User model subroutine name (* , 3) - User report subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector (see Local Work Arrays)
NSIZE	I	INTEGER	—	Length of size results vector
SIZE	O	REAL*8	NSIZE	Real sizing results (see Size)
INTSIZ	O	INTEGER	NSIZE	Integer size parameters (see Size)
LD	I	INTEGER	—	Plex location of the stream class descriptor bead

[†]I = Input to subroutine, O = Output from subroutine, W = Workspace

Stream Structure and Calculation Sequence

The stream structure for material streams is described in Appendix C. For information streams, the stream vector consists of a single value. All stream data are in SI units.

In normal sequential modular calculations, the block calculates (fills in) the outlet streams based on the inlet streams and specifications you made for the block. However, sometimes it is useful for a block to modify its feed stream(s), such as when the feed stream has no source block because it is a feed to the process. To handle these cases, the user subroutine may change its inlet stream.

The list of inlet streams for User2 blocks includes streams referenced on the **Streams** sheet, in addition to streams connected on the flowsheet.

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed if property calculations are to be performed using the first (or only) option set. NBOPST(1,2) should be passed if property calculations are to be performed using the second option set.

Size

The size array SIZE should be filled in with the results from simulation calculations if sizing and costing calculations are performed. The values that are stored depend on the cost model. The values can be stored in any order. For each element in SIZE, the corresponding element in INTSIZ should be filled out with the correct result code. The result codes are:

10	Inlet pressure (N/m ²)
11	Outlet pressure (N/m ²)
12	Inlet temperature (K)
13	Outlet temperature (K)
14	Inlet vapor volumetric flow (m ³ /s)
15	Outlet vapor volumetric flow (m ³ /s)
16	Outlet liquid volumetric flow (m ³ /s)
17	Vapor density (kg/m ³)
18	Liquid density (kg/m ³)
19	Heat duty (watt)
50	Area (m ²)
51	Isentropic or polytropic efficiency
52	Cp/Cv
53	Mechanical efficiency
54	Compressor type (1=polytropic centrifugal; 2=polytropic positive displacement; 3=isentropic)
55	Horsepower (watt)
92	Surface tension (N/m)
93	Liquid viscosity (N-s/m ²)
94	Relative volatility
95	Diffusivity (m ² /s)

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **User | Input | Specifications** sheet or the **User2 | Input | User Arrays** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters, and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **User** or **User2 | Input | Specifications** sheet. Aspen Plus does not retain these arrays from one call to the next.

Simulation Control Guidelines

An Aspen Plus unit operation model calculates outlet stream values, and real and integer array results, from inlet stream values and from real and integer array inputs, in order to produce History file and Report file output.

Follow these guidelines to implement the simulation options in a user unit operation model:

- 1** If USER_NGBAL of COMMON /PPEXEC_USER/ is 1, the energy balance switch is on; and if USER_IPASS of COMMON /PPEXEC_USER/ is 1, the model is being called on a simulation pass and should calculate:
 - o Outlet stream component flows
 - o Total flow
 - o Molecular weight
 - o Pressure
 - o Enthalpy (specific mass enthalpy in Aspen Plus streams)
 - o Retention variables in the integer and real arrays
- 2** If USER_NGBAL of COMMON /PPEXEC_USER/ is 1, the energy balance switch is on; and if USER_IPASS of COMMON /PPEXEC_USER/ is 2, the model is being called on a results pass and should fill in values of the outlet streams and of the integer and real arrays that are not calculated during a simulation pass. The values that were calculated during a simulation pass should not be changed.
- 3** If USER_NGBAL of COMMON /PPEXEC_USER/ is 1, the energy balance switch is on; and if USER_IPASS of COMMON /PPEXEC_USER/ is 3, then the model is being called on a simulation and results pass and should calculate everything in both guidelines 1 and 2 above.
- 4** A sizing flag is stored in USER_ISIZE of COMMON /PPEXEC_USER/. When the value of USER_ISIZE is:
 - 0 - Skip sizing calculations
 - 1 - Do sizing calculations
- 5** USER_NGBAL, USER_IPASS and USER_ISIZE are set by the system before each User or User2 block call. Their values depend on the simulation options you specified on the **Setup | Calculation Options | Calculations** sheet, or on the **User** or **User2 | Block Options | Simulation Options** sheet. USER_IPASS also depends on the progress through the simulation, results, and report passes. A USER_IPASS of 4 is the report pass.
- 6** The restart flag is stored in USER_IRESTR of COMMON /PPEXEC_USER/. USER_IRESTR indicates whether initialization calculations are to be performed or whether the model should simply restart using retention information. The initial value of the restart flag depends on the option you specified on the **Setup | Calculation Options | Calculations** sheet, or

on the User BlockOptions SimulationOptions (or **User** or **User2 | Block Options | Simulation Options**) sheet.

- 7 When the value of USER_IRESTR is:
 - 0 - Initialization calculations should be performed.
 - 1 - Initialization calculations should be performed. If the model calculations proceed satisfactorily, retention information should be stored and USER_IRESTR should be set to 2.
 - 2 - Initialization should be bypassed, and the model should restart iterations using retention information. Retention information should be updated. If for any reason you consider the new retention information unreliable and do not want to use it to restart calculations, reset USER_IRESTR to 1.
- 8 The simulation options discussed above eliminate unnecessary process calculations. However, since they complicate model coding, in many cases you can ignore them. For example:
 - o If a model always calculates outlet stream enthalpy, you can ignore the energy balance switch USER_NGBAL.
 - o If all results are calculated every time the model is called, you can ignore the USER_IPASS distinctions.
 - o If a model cannot restart from retention information, you can ignore the restart flag, USER_IRESTR.
 - o If a model does not do sizing calculations, you can ignore USER_ISIZE.
- 9 Unit operation models also set the convergence flag, USER_ICONVG, of COMMON /PPEXEC_USER/. If the model calculations have completed normally (converged), USER_ICONVG should be set to 0. If the model calculations have failed, USER_ICONVG should be set to a negative number. The value of the negative number should indicate the type or cause of failure so that an appropriate message is written during the report pass.
- 10 The following values of USER_ICONVG are reserved for Aspen Plus system use and should not be used by the User or User2 model:
 - o -1
 - o -2
 - o -9995
 - o -9996
 - o -9997
 - o -9998
 - o -9999
 - o -USER_IUMISS

History File

The History File contains intermediate output during unit operation model execution. History File output consists of errors, warnings, and diagnostics. You can control the amount of History File output by setting a message level on the **Setup | Specifications | Diagnostics** sheet and the **User** or **User2 | Block Options | Diagnostics** sheet. Each level includes the messages written at all lower levels. The message level for a User or User2 block is

stored in USER_LMSG of COMMON /PPEXEC_USER/. The Fortran unit number for the History File is stored in USER_NHSTRY of COMMON /PPEXEC_USER/.

Note: WRITES to this file will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

At all diagnostic levels of 4 or above, Aspen Plus writes a message to the History File when the unit block begins execution. The user model then writes the errors, warnings, and diagnostics.

Diagnostics should begin in column seven or beyond. Units are assumed to be SI. If they are not SI, unit labels should accompany numerical values.

Errors and warnings have a standard format (see Aspen Plus Error Handler, Chapter 4).

Terminal File

If you want to write to the terminal or the Control Panel, use the Terminal File Writer utility DMS_WRTTRM (see Chapter 4).

Report File

The Report file Fortran unit number is stored in USER_NRPT of COMMON /PPEXEC_USER/ (see Appendix A; and Report Header Utility, Chapter 4).

Note: WRITES to this file will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

Control Panel

Aspen Plus displays on the Control Panel runtime information and any errors or warnings that occur during calculations. The Control Panel replaces the Terminal File for output purposes. Use the utility DMS_WRTTRM to write to the Control Panel (see Chapter 4).

You cannot use the Control Panel to read from the Terminal File. In this case, you will need to run Aspen Plus without the Windows user interface. The Fortran file number for the terminal is stored in USER_NTRMNL of COMMON/PPEXEC_USER/ (see Appendix A).

Incorporating Excel Worksheets into User2

If you are one of the many engineers who use Excel models, you may want to:

- Combine Excel models with the Aspen Plus system.
- Supply input data and additional operating parameters to your Excel model through the Aspen Plus graphical user interface, and then retrieve results.

This section explains how to run Excel models as part of an Aspen Plus simulation, without having to convert the Excel model into a Fortran library. This section assumes you are familiar with the Aspen Plus User2 model, described earlier in this chapter.

Extending the User2 Concept

There are four steps in extending the User2 model to incorporate Excel models:

- 1 Place a User2 block.
- 2 Connect any streams.
- 3 Specify the name of the Excel workbook file that represents the unit operation model for the block.
- 4 Enter the operating parameters, by filling out the real and integer parameters arrays.

When Aspen Plus solves the block, it loads the Excel file, writes the appropriate input data, tells the workbook to calculate, and then reads back the output data. The integer and real parameter arrays are treated as both input and output.

Excel File Name

The file name for the Excel model is limited to 64 characters. All characters in the file name must be alpha or numeric, a hyphen (-), or an underscore; spaces are not allowed. The file name should be the full path to the file. The .xls format from Excel 2003 should be used, not the .xlsx, .xlsm, or .xlsb formats from later versions. For example:

```
e:\mymodels\myflash.xls
```

Fortran Routine

In addition to the file name, you may optionally enter the name of the Fortran routine to call when solving the User2 block. If you don't enter a Fortran routine, then a default routine will be used.

In the default routine, all the inlet stream data (component flows, stream total flow, temperature, pressure, mass-specific enthalpy, molar vapor fraction, molar liquid fraction, mass-specific entropy, mass density and molecular weight for the stream) are initially copied to a single array. Then the appropriate APIs are called, to write and read from the Excel workbook. The outlet stream data is copied into the outlet stream array for Aspen Plus to continue the simulation. All the stream data that is transferred between Aspen Plus and Excel must be in SI units (kmole/sec, K, Pascal).

The Excel Template

To be compatible with the Aspen Plus User2 extension, the workbook representing a model must follow certain guidelines. These guidelines are called the Excel Template.

The template, `userxltemplate.xls`, is located in the **user** directory of the Aspen Plus engine installation.

To begin creating an Excel model, copy the template.

Tables

In the Excel template, data is divided into four distinct tables. Each table is on its own sheet. The sheet and the corresponding table have the same name.

The **Aspen_Input** table supplies the stream input information, which includes:

- Flow for each component in the stream (MIXED substream).
- Temperature, pressure and total flow for the stream (MIXED substream).
- Mass-specific enthalpy, molar vapor fraction, molar liquid fraction, mass-specific entropy, mass density and molecular weight for the stream (MIXED substream).

The columns for the table are for inlet stream IDs. The rows display the component IDs followed by total mole flow, temperature, pressure, mass-specific enthalpy, molar vapor fraction, molar liquid fraction, mass-specific entropy, mass density and molecular weight for the respective stream (MIXED substream). Whenever there is no other column or row heading available, the system automatically uses the column or row number as the heading.

In the sample workbook called `dryer.xls`, the **Aspen_Input** sheet looks like:

INPUT	2	H2O-COAL
AIR	0.0043	0
H2O	0	0.002182
COAL	0	0.001818
TOTFLOW	0.0043	0.004
TEMP	478	300
PRES	101325	101325
ENTHALPY	181423	-3479200
VAP FRAC	1	0
LIQ FRAC	0	1
ENTROPY	475.7966	11366.47
DENSITY	0.7381	725.183
MOLE WT	28.95091	48.87867

The **Aspen_RealParams** table shows the array of real parameters as entered on the User2 input form in Aspen Plus. Whenever there is no other column or row heading available, the system automatically uses the column or row number as the heading:

REALPARAMS	1
1	0.1
2	0.3

The **Aspen_IntParams** table shows the array of integer parameters as entered on the User2 input form. Whenever there is no other column or row

heading available, the system automatically uses the column or row number as the heading:

INTPARAMS	1
1	1

The **Aspen_Output** table shows the output data for each stream. The types of data correspond to those in the Aspen_Input table. The columns for the table are for outlet stream IDs. Rows display the component IDs followed by total mole flow, temperature, pressure, mass-specific enthalpy, molar vapor fraction, molar liquid fraction, mass-specific entropy, mass density and molecular weight for the respective stream (MIXED substream):

OUTPUT	AIROUT	PRODUCT
AIR	0.0043	0
H2O	0.00086	0.001322
COAL	0	0.001818
TOTFLOW	0.00516	0.00314
TEMP	478	300
PRES	101325	101325
ENTHALPY	0	0
VAP FRAC	0	0
LIQ FRAC	0	0
ENTROPY	0	0
DENSITY	0	0
MOLE WT	0	0

Other tables may appear if you have solid substreams in streams connected to this model, such as **Aspen_Input_NC** and **Aspen_Output_NC** for an NC substream. These are similar in form to the ones shown above, but have substream and component attributes added at the end of the list.

The data in each table is given a named range, with the same name as its sheet. By naming the ranges, models can be written by referring to these ranges, rather than using explicit cell references. This technique allows for more readable formulas and macros, and is more robust and flexible under change. For example, in the dryer Sheet1, the input pressure for the second stream is referred to using the range name Aspen_Input in the following formula, which uses the helper function ahgetvalue (see next section):

```
=ahgetvalue(Aspen_Input, "PRES", 2)
```

The Helper Functions

To facilitate extracting data from the tables, a set of helper functions is supplied. All the functions are prefixed with "ah", which stands for Aspen Helper. These functions are designed to be used both from within formulas in cells, and from within VBA functions and subroutines. The functions are as follows:

AhGetValue

```
Public Function AhGetValue(R As Range, Row As Variant, Optional  
Col As Variant) as Variant
```

This function retrieves the value of the cell specified by the column and row numbers of a range. The return value will normally be a number or a string.

R is the range of the cell specified by the row and column. When the function is used on the formula bar of a worksheet, it can be written as:

```
ahGetValue(RangeName, ...).
```

When the function is used in Visual Basic, the range can be in the form of either:

```
ahGetValue([RangeName], ...)
```

Or

```
ahGetValue(Range("RangeName"), ...).
```

Row and **Col** are the column and row numbers of the cell whose value is retrieved. Both can be passed as index numbers, or as the names of the components or streams. For example, if AIR is the first input stream of a model, and H2O is the second input component of stream AIR, then AIR is in column 1, and H2O is in row 2. To retrieve the value of H2O in AIR, write the function in any of the following forms:

```
ahGetValue([aspen_input], "H2O", "AIR")  
ahGetValue([aspen_input], 2, "AIR"),  
ahGetValue([aspen_input], "H2O", 1)  
ahGetValue([aspen_input], 2, 1)
```

Column number is optional. If you do not specify column number, its default value is 1.

For example, this call returns the value of the H2O component of the AIR input stream:

```
ahGetValue([Aspen_Input], "H2O", "AIR").
```

AhSetValue

```
Public Sub ahSetValue(R As Range, Row As Variant, Col As Variant,  
VNewValue As Variant)
```

This sub-procedure sets the value of the cell specified by the column and row numbers of a range.

R is the range of the cell specified by the row and column. When the function is used on the formula bar of a worksheet, you can write it as:

```
ahSetValue(RangeName, ...)
```

When the function is used in Visual Basic, the range can be in the form of either:

```
ahSetValue([RangeName], ...)
```

Or

```
ahSetValue(Range("RangeName"), ...).
```


Row and **Col** are the column and row numbers of the cell whose value is set. Both can be passed as index numbers, or the names of the components or streams. For example, if AIR is the first input stream of a model, and H2O is the second input component of stream AIR, then AIR is in column 1, and H2O is in row 2. To set the value of H2O in AIR, write the function in any of the following forms:

```
ahSetValue([Aspen_Output], "H2O", "AIR", ...)
ahSetValue([Aspen_Output], 2, "AIR", ...)
ahSetValue([Aspen_Output], "H2O", 1, ...)
ahGetValue([Aspen_Output], 2, 1, ...).
```

VNewValue is the value to be set; it can be number, string, or other data types. For example, this call sets the value of H2O of the input stream AIR to 0.004:

```
ahSetValue([Aspen_Output], "H2O", "AIR", 0.004)
```

AhNumComps

```
Public Function ahNumComps(R As Range) As Integer
```

This function calculates the number of components of a model.

R is the range that includes all the streams and components of a model. When the function is used on the formula bar of a worksheet, you can write it as:

```
AhNumComps (RangeName)
```

When the function is used in Visual Basic, the range can be in either of the following two forms:

```
AhNumComps ([RangeName])
AhNumComps (Range ("RangeName"))
```

For example, this call returns the number of input components:

```
ahNumComps ([Aspen_Input])
```

AhNumStreams

```
Public Function ahNumStreams(R As Range) As Integer
```

This function calculates the number of streams of a model.

R is the range that includes all the streams and components of a model. When the function is used on the formula bar of a worksheet, you can write it as:

```
ahNumStreams (RangeName)
```

When the function is used in Visual Basic, the range can be in either of these two forms:

```
ahNumStreams ([RangeName])
ahNumStreams (Range ("RangeName")).
```

For example, this call returns the number of input streams:

```
ahNumStreams ([Aspen_Input])
```

AhNumParams

```
Public Function ahNumParams(R As Range) As Integer
```

This function returns the number of integer or real parameters in a model.

R is the range that represents the parameters of a model, usually Aspen_RealParams or Aspen_IntParams. When the function is used on the formula bar of a worksheet, you can write it as:

```
ahNumParams(RangeName)
```

When the function is used in Visual Basic, the range can be in either of these two forms:

```
ahNumParams ([RangeName])  
ahNumParams (Range("RangeName"))
```

For example, this call returns the number of real parameters:

```
ahNumParams([Aspen_RealParams])
```

The Hook Functions

When calling some of the User2 routines, Aspen Plus calls pre-defined macros in the Excel file. These Excel macros are called the hook functions. They allow the modeler to customize and extend the basic sequence in running an Excel model.

When the Aspen Plus engine processes a User2 Excel block, it calls the following APIs from the User2 interface file:

```
StartIteration(fileName, blockID)  
WriteTable("Input". ...)   
WriteTable("IntParams". ...)   
WriteTable("RealParams". ...)   
WriteTable("Output". ...)   
CalculateData()  
ReadTable("IntParams". ...)   
ReadTable("RealParams". ...)   
ReadTable("Output". ...)   
EndIteration
```

When the entire simulation run has completed, Aspen Plus calls **EndRun(runID)**.

Function AspenStartIteration(blockId As String) As String

This function is called at the start of each iteration of the Excel model, before the model gets calculated. A global variable, g_blockId, is assigned to the current block ID, so that it can be used in AspenEndIteration or in other methods.

Function AspenEndIteration() As String

This function is called at the end of each iteration of the model, after the model has been calculated. If you want to save the last table of data to a uniquely named sheet, this is the place to do it. The Excel templates contain a function called CopySheetForBlock. This function takes a sheet name (such as Output or Input) and a block ID. The template makes a call to:

```
CopySheetForBlock "Output", g_blockId
```

This call copies whatever values are currently on the Aspen_Output sheet onto a newly created sheet called Aspen_Output_BlockId, where BlockId is the ID of the block that is ending its iteration. The g_blockId gets saved during the call to AspenStartIteration.

Function AspenCalculate() As String

This function is called to solve the model for the given inputs, after writing out all the input data. By default it calls only Calculate. If you are writing VBA code to solve your model, call it from here.

Function AspenEndRun(runId As String) As String

This function is called when the Aspen Plus engine is quitting, after all blocks have been processed. The run ID is passed from the engine. You may want to use the run ID as part of the file name, if saving the sheet at the end of a run.

Error Handling

Each of the hook functions returns a string. If the string is empty, that indicates success. A non-empty string indicates failure; the value of the string is passed back to the engine, and then to the user. For example, if AspenCalculate() returns the string "Pressure exceeds maximum," this information will be passed back to the user and an error status will be displayed.

The Sample Workbook

A sample workbook called: dryer95.xls or dryer.xls is located in the **user** directory of the Aspen Plus engine installation. This workbook is an example of a model designed for a specific set of components and streams. It simulates a model that removes water from a slurry stream, based on an evaporation rate derived from the mole fraction of the input and output air streams. These mole fractions are passed to the model via the real parameters array, which is written out to the Aspen_RealParams sheet.

In this sample you will see five sheets of interest: Aspen_Input, Aspen_Output, Aspen_Real Params and Sheet1. There are named ranges for each of the Aspen sheets. The cell containing the evaporation rate is named EvapRate. EvapRate is defined on Sheet1 as follows:

```
ahgetvalue(Aspen_Input,"TOTFLOW",1) *  
(ahgetvalue(Aspen_RealParams,2) - ahgetvalue(Aspen_RealParams,1))
```

EvapRate is then referred to on the Aspen_Output sheet to calculate the water flow for the air and slurry streams. The rest of the values are just taken from the input sheet, via the ahGetValue method. Because this model is written exclusively for the given set of streams and components, cell formulas can be placed directly in the Aspen_Output range.

When a new simulation is run, the input data and real parameter data will be overwritten with new values from the input file passed to the Aspen Plus engine. The output sheet will not be overwritten, because cells that contain formulas are never overwritten.

At the end of each solving iteration, the output values will be read back into the engine. The last output values will be saved to a sheet, based on the name of the block. At the end the entire workbook is saved via a call to ThisWorkbook.Save, made from the hook function AspenEndRun.

A sample function called testVBACalculate produces the same output table data as the formulas, but does so from VBA code. It makes extensive use of the ahSetValue() method.

Creating or Converting Your Own Excel Models

Start by copying the template, userxltemplate.xls, located in the **user** directory of the Aspen Plus engine installation.

Decide whether you are going to write a generic or specific model, as explained later in this chapter.

To create sample data, you may need to extend one of the sample named ranges: Aspen_Input, Aspen_Ouput, or one of the parameter ranges. Use the Insert/Name/Define dialog to modify the current range. Then enter appropriate sample stream and component names in place of the S1, S2,... and C1, C2 that are supplied. You can test your model as you develop it, by running the AspenCalculate macro or just recalculating your workbook. Once your model is fully tested from within Excel, then you can begin testing it from Aspen Plus.

Converting Existing Formulas

In your spreadsheet where you refer to the temperature of an inlet stream called AIR, substitute the ahGetValue(Aspen_Input,"TEMP","AIR"). Do likewise for your other input. You can place your output formulas directly onto the Aspen_Output sheet when writing a specific model. By using the helper methods, you will be making your formulas more readable and more robust as you make modifications to the model.

Writing Models for a Specific Set of Components and Streams

When you are writing a model that is always going to be used with the same set of components and streams, you can embed formulas into your worksheet, and refer to streams and components by name, making your model much easier to read. For example, to get the water flow of the air input stream you can write:

```
ahGetValue(Aspen_Input, "H2O", "AIR")
```

When data is written to the spreadsheet, cells that contain formulas are never overwritten. All cells that don't contain formulas are cleared before new data is written.

Writing Generic Models in Excel

A generic model, like the built-in Aspen Plus unit operation models, doesn't make assumptions about the stream composition, the number of streams, or their names. The best way to create such a model in Excel is to use the Aspen helper functions to determine the numbers of streams and components, and iterate through them. You can then write your calculation routines in the Aspen hook method `AspenCalculate()`, or call your methods from there. Included in the Excel template are some test and sample code in `ahTest()`, which show how to call the various "ahGet" methods. The following code snippet loops through all the component flows of all the input streams:

```
For i = 1 To ahNumStreams([Aspen_Input])
  For j = 1 To ahNumComps([Aspen_Input])
    testSheet.Cells(rowNum+j,
i+1).Value=ahGetValue([Aspen_Input], j, i)
  Next j
Next i
```

Converting an Existing Excel Model

The procedure for converting an Excel model is:

- 1 Copy to the working directory the Aspen Excel template `userxltemplate.xls` from the **user** directory of the Aspen Plus engine installation. Copy the template to a file that represents your model; for example, `mymodel.xls`.
- 2 For the user Excel sheets that have formulas associated with the cells, copy Sheet1 of the user Excel sheet for the model you are converting to the Aspen Excel template (also called Sheet1).
- 3 Start up the Aspen Plus user interface, `apwn.exe`. Place a User2 block and make the appropriate stream connections.
- 4 Complete the input specifications, including the User2 input form (Excel file name and Fortran interface file). Specify the Aspen Excel template name (for example, `mymodel.xls`) on the User2 input form.
- 5 Run the simulation.
- 6 Bring up `mymodel.xls`.
- 7 Aspen_Input and Aspen_Output sheets will contain the appropriate stream and component IDs for the simulation you are modeling.

- 8 Go to Sheet1 of mymodel.xls. Use the helper functions to retrieve appropriate values from the Aspen_Input sheet. Identify the INT and REAL parameters that need to be entered from Aspen Plus, and any results that need to come back into Aspen Plus. Use helper functions to retrieve the INT and REAL values from the Aspen_IntParams sheet and the Aspen_RealParams sheet.
- 9 Go to the Aspen_Output sheet of the Aspen Excel template. Retrieve the stream results from Sheet1. For example, if you want to retrieve the value in cell c12 on Sheet1, you can type in: =Sheet1!C12
- 10 If you have any results to be stored in INT or REAL arrays, go to the Aspen_IntParams and Aspen_RealParams sheets. Retrieve those values from Sheet1 as described in Step 8.

You are now ready to run the simulation.

Customizing the Fortran Code

The default file `userxls.f` is supplied with the product. It allows you access to a commonly used subset of model data. To extend the amount of data that gets placed into tables in Excel, you can modify this Fortran file, and make use of the same `TableDataWrapper` API functions. The API is implemented in the `TableDataWrapper.dll`. All the API methods take a return code as their first argument. This return code will be 0 on success, and non-zero on failure. Details about the error can be retrieved by calling `GetLastTableDataErr()`.

StartIteration()

Description

When you start Iteration for a block, the unique block name of the block for which the calculations are to be done is retrieved, along with the file name of the Excel workbook that represents the model.

For each iteration of a block, a hook called `AspenStartIteration(blockId)` will be called in the Excel workbook, giving you a chance to extend the model.

Syntax

```
void StartIteration(FORINT* retCode, HOLLSTR filename, FORINT
filenameLen, HOLLSTR blockid, FORINT blockidLen)
```

Arguments

filename	Name of the Excel file that represents this model. Will normally come from the User2 input form.
filenameLen	Number of characters in the filename string.
blockid	The unique block name of the block being calculated.
blockidLen	The number of characters in the block name string.

WriteTable()

Description

For each set of data, such as parameters containing INT and REAL, or a Stream table containing a table of components and Streams, the following are passed as input parameters:

- Number of items in the row and col.
- Row and col headers.

You need one WriteTable call for each set of data. Create new Excel sheets for each table of data, if they don't already exist. Assign the tag "Aspen_" , pre-pended to the table name. For example, for a table called Input the sheet name would be Aspen_Input.

Use a separate worksheet for each set or table of data. The four tables used in the supplied usrxls.f are Input, Output, RealParams and IntParams. These are the main working sheets, where the input data is written, calculations (if any) occur, and the resulting data is held.

Before data is written, the whole range is cleared, except for those cells that contain formulas.

The data is written starting from the top left corner of the sheet. As a block of data is being filled into the sheet, the data corresponding to those cells that need a blank are indicated by one of the following:

This constant	Represents
IMISS	Integer missing value
RMISS	Real missing value

Syntax

```
void WriteTable (FORINT* retCode, HOLLSTR tableName, FORINT*
tableNameLen, FORINT* numRows, HOLLSTR rowheaders, FORINT*
rowheaderLen, FORINT* numcols, HOLLSTR colheaders, FORINT*
colheaderLen, FORINT* dataType, void* data, FORINT* dataLength);
```

Arguments

tableName	The name of the overall table or range, input, params, output.
tableNameLen	The number of characters in the tablename string.
numrows	The number of rows in the table.
rowheaders	An array of strings representing the headers for each row. All the strings must be of the same length. Either the array must be numRows big, or the rowheaderLen must be 0.
rowheaderLen	The number of characters in the each string of the rowheaders array. If this number is 0, then the row names for the table will be automatically generated as 1, 2, 3, ... up to numRows.
numcols, colheaders, colheaderLen	See the preceding description of numRows, rowheaders, and rowheaderLen.
dataType	Indicates what type of data is being passed in the data array. 0 means integer, 1 means real, and 2 means string.

data	The actual data to be written into the body of the table. It is read column by column, and must be as large as numRows * numcols.
dataLength	If the dataType is string, then the dataLength holds the number of characters in the fixed len strings in data.

CalculateData

Description

Calls a hook function in the Excel workbook called AspenCalculate to perform the calculations. Users can call their own VBA calculation macros from AspenCalculate.

Syntax

```
void CalculateData(FORINT*retCode)
```

ReadTable

Description

Read the results data and other blocks of input data that were subject to change during calculation. For each table of data to be read, such as parameters containing INT and REAL, the details such as the number of items in the rows and cols are passed in as input parameters.

Syntax

```
void ReadTable(FORINT* retCode, HOLLSTR tableName, FORINT*
tableNameLen, FORINT* numRows, FORINT* numcols, FORINT* dataType,
void* data, FORINT*dataLength);
```

Arguments

tableName	The name of the overall table or range, input, params, output.
tableNameLen	The number of characters in the tablename string.
numRows	The number of rows in the table.
numcols	The number of columns in the table.
dataType	Indicates what type of data is being passed in the data array. 0 means integer, 1 means real, and 2 means string.
data	The actual data to be read from the body of the table. It is read column by column, and must be as large as numRows * numcols.
dataLength	If the dataType is string, then the dataLength holds the number of characters in the fixed Len strings in data.

EndIteration

Description

Ends the iteration for a block. Calls a hook function called `AspenEndIteration`. This hook allows the modeler to do things like copying the table data of the Active block to the block's specific table sheet. For example, the Input sheet of the block B1 could be copied to `Aspen_Input_B1` sheet. During this process only the values in the sheet, and not the formulas and the name ranges, are copied.

Syntax

```
void EndIteration(FORINT* retCode)
```

GetLastTableDataErr

Description

Gets the last error, if any, from any of the API calls.

Syntax

```
void GetLastTableDataErr(FORINT* retCode, FORINT*errNumber,  
HOLLSTR description, FORINT*descriptionLen, FORINT*  
descriptionLineWrapLen, HOLLSTR source, FORINT* sourceLen)
```

Arguments

errNumber	The error number of the last error. It may be an automation error, an Excel error, or a table wrap error.
description	The error description as a string.
descriptionLen	On the way in, this parameter holds the maximum number of characters that can be stored in the description string. When returning, it holds the actual number of characters written in the description string.
descriptionLineWrapLen	If this parameter is non-zero, the description string will be formatted so that it breaks cleanly at lines <code>descriptionLineWrapLen</code> long, padding the lines with spaces.
source	The name of the application that is the source of the error.
sourceLen	The maximum number of characters that can be written to the source string.

Accessing User2 Parameters

The following subroutines and functions can be called from a User2 Fortran subroutine to access the real, integer, and character parameter arrays specified on the User2 Setup User Arrays sheet, and the names of the Excel interface file and additional user subroutines specified on the User2 Setup Subroutine sheet. There are 3 ways to get or set the values of these parameters:

- 1 Access the values directly from the INT and REAL arrays (for example, INT(5)).
- 2 Use the helper functions that access the parameters by position.
- 3 Use the helper functions that access the parameters by name and index, using the names on the **User2 | Setup | Configured Variables** sheet. See *Getting Started Customizing Unit Operation Models*, Chapter 4, to learn how to use the **Configured Variables** sheet.

Accessing parameters by position

This group of subroutines (USRUTL_NUMSTR through USRUTL_GETEXCEL) can be called from User2 without using the names in the user configuration subroutine. They give the user access to integer, real, and character parameter arrays. Access to the arrays is by position, not by name.

```
SUBROUTINE USRUTL_NUMSTR (NUMSTR)
```

Sets NUMSTR to the number of character string parameters.

```
SUBROUTINE USRUTL_GETHSTR (ISTRNO, IBUFF, IBUFLEN, IACTLEN)
```

Retrieves character parameter number ISTRNO as a Hollerith array in IBUFF. Sets IACTLEN to the actual length of the string, in characters. The user must set IBUFLEN to the length of the integer array IBUFF before calling this subroutine.

```
SUBROUTINE USRUTL_GETCSTR (ISTRNO, CSTR, CSTRLEN, IACTLEN)
```

Retrieves character parameter number ISTRNO as a character array in CSTR. Sets IACTLEN to the actual length of the string, in characters. If the user sets CSTRLEN to zero before calling this routine, then the only action is to set IACTLEN. If CSTRLEN is greater than zero, then CSTR is filled and IACTLEN is set to the number of characters in CSTR.

```
SUBROUTINE USRUTL_SETHSTR (ISTRNO, IBUFF, ICLEN)
```

Sets character parameter number ISTRNO from the first ICLEN characters of Hollerith array IBUFF.

```
SUBROUTINE USRUTL_SETCSTR (ISTRNO, CSTR, ICLEN)
```

Sets character parameter number ISTRNO from the first ICLEN characters of character array CSTR.

SUBROUTINE USRUTL_NUMINTS (NUMINTS)

Retrieves the length of the integer parameter array as NUMINTS.

SUBROUTINE USRUTL_GETINT (INTNO, IVAL)

Sets IVAL to the value of integer parameter number INTNO.

SUBROUTINE USRUTL_SETINT (INTNO, IVAL)

Sets value number INTNO in the integer parameter array to value IVAL.

SUBROUTINE USRUTL_NUMREALS (NUMREALS)

Retrieves the length of the real parameter array.

SUBROUTINE USRUTL_GETREAL (IREALNO, RVAL)

Sets RVAL to the value of real parameter number IREALNO.

SUBROUTINE USRUTL_SETREAL (IREALNO, RVAL)

Sets value number IREALNO in the real parameter array to value RVAL.

Argument List Descriptions for Position-Access Helper Functions

Variable	I/O Type		Dimension	Description
NUMSTR	O	INTEGER	—	Number of character strings
ISTRNO	I	INTEGER	—	1-based string index
IBUFF	I/O	INTEGER	IBUFLEN, when output	Hollerith string as array of integers, 4 chars/word
IBUFLEN	I	INTEGER	—	Length of IBUFF
ICLEN	I	INTEGER	—	Number of characters (not words) to copy
IACTLEN	O	INTEGER	—	Number of characters in retrieved string
CSTR	I/O	CHARACTER	CSTRLEN, when output	Character string
CSTRLEN	I	INTEGER	—	Length of CSTR. If 0, only set IACTLEN.
NUMINTS	O	INTEGER	—	Number of integer values entered
INTNO	I	INTEGER	—	1-based integer index
IVAL	I/O	INTEGER	—	Integer value
NUMREALS	O	INTEGER	—	Number of real values entered
IREALNO	I	INTEGER	—	1-based real index
RVAL	I/O	REAL*8	—	Real value

Accessing parameters by name

The following group of functions allows the user to set and get the values from the integer, real, and character arrays by name. The value of VNAME should be set to the name of a variable in the user configuration subroutine. The INDEX array is always 1-based.

The **User2 | Setup | Configured Variables** sheet allows the user to set values in the integer, real, and character arrays using the names specified in

the configuration library. The values are actually the values in those arrays on the **User2 | Setup | User Arrays** sheet. When a value is set on the **User Arrays** sheet, the corresponding value is updated on the **Configured Variables** sheet, and vice versa. See *Getting Started Customizing Unit Operation Models*, Chapter 4, to learn how to use the **Configured Variables** sheet.

```
INTEGER FUNCTION USRUTL_GETVAR (VNAME, INDEX)
```

The returned value is the 1-based location in the integer or real array of the integer, real, or character variable with name VNAME and 1-based indices specified by array INDEX (a scalar value may be entered for scalar and one-dimensional variables). Character variables are stored in the integer parameter array in Hollerith format (4 characters per integer word).

```
INTEGER FUNCTION USRUTL_GET_REAL_PARAM (VNAME, INDEX, RVAL)
```

Sets RVAL to the value of the real parameter with name VNAME and indices as specified in INDEX. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_GET_INT_PARAM (VNAME, INDEX, IVAL)
```

Sets IVAL to the value of the integer parameter with name VNAME and indices as specified in INDEX. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_GET_CHAR_PARAM (VNAME, INDEX, CHARSTR, IACTLEN)
```

Fills the character array CHARSTR with the value of the character parameter specified by VNAME and INDEX. Sets IACTLEN to the actual length of the string, in characters. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_GET_HCHAR_PARAM (VNAME, INDEX, ICHARHOL, ICHLEN, IACTLEN)
```

Fills the integer array ICHARHOL with the Hollerith representation (4 characters per integer) of the character parameter specified by VNAME and INDEX. Sets IACTLEN to the actual length of the string, in characters. The user must set ICHLEN to the length of the integer array ICHARHOL. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_SET_REAL_PARAM (VNAME, INDEX, RVAL)
```

Sets the value of the real parameter with name VNAME and indices as specified in INDEX to value RVAL. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_SET_INT_PARAM (VNAME, INDEX, IVAL)
```

Sets the value of the integer parameter with name VNAME and indices as specified in INDEX to value IVAL. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_SET_CHAR_PARAM (VNAME, INDEX, CHARSTR)
```

Sets the character parameter specified by VNAME and INDEX to the character string CHARSTR. Returns GLOBAL_IMISS on any error, 0 otherwise.

```
INTEGER FUNCTION USRUTL_SET_HCHAR_PARAM (VNAME, INDEX, ICHARHOL, ICHLEN)
```

Sets the character parameter specified by VNAME and INDEX to the Hollerith string ICHARHOL. User should set ICLEN to the number of characters to copy. Returns GLOBAL_IMISS on any error, 0 otherwise.

Argument List Descriptions for Name-Access Helper Functions

Variable	I/O	Type	Dimension	Description
VNAME	I	CHARACTER	*(*)	Name of var to find
INDEX	I	INTEGER	varies	Array of indices, 1-based
RVAL	I/O	REAL*8	—	Real value
IVAL	I/O	INTEGER	—	Integer value
CHARSTR	I/O	CHARACTER	IACTLEN, when output	Character string
IACTLEN	O	INTEGER	—	Number of characters in CHARSTR
ICHARHOL	I/O	INTEGER	ICHLEN, when output	Hollerith array
ICHLEN	I	INTEGER	—	Length of ICHARHOL (integer words)
ICLEN	I	INTEGER	—	Number of characters to store

Other Helper Functions

These subroutines give access to the Excel file name and the user subroutines named on the USER-MODELS sentence, and allow the user to convert between Hollerith strings and character strings. Each of the subroutines is described below.

```
SUBROUTINE USRUTL_HTOCHAR ( IARRAY, IARRAYLEN, LOC, CHARSTR )
```

Converts the Hollerith character string at location LOC in the array IARRAY into a character string variable stored in CHARSTR. The user should set IARRAYLEN to the length of IARRAY.

```
SUBROUTINE USRUTL_CHARTOH ( CHARSTR, ICHARHOL, ICHLEN )
```

Converts character string CHARSTR to a Hollerith string and stores it in ICHARHOL. The user should set ICHLEN to the length of ICHARHOL.

```
SUBROUTINE USRUTL_GETEXCEL ( INAME, NAMELEN )
```

Retrieves the name of the Excel spreadsheet as a Hollerith string in INAME. Sets NAMELEN to the number of characters in the Excel spreadsheet name. INAME should be an integer array of length at least 40.

Use INAME as an input argument for the StartIteration function called by the Excel interface user model subroutine (see Customizing the Fortran Code, in this chapter).

```
SUBROUTINE USRUTL_NUMSUBS ( NUMSUBS )
```

Retrieves the number of user subroutines entered on the USER-MODELS sentence.

```
SUBROUTINE USRUTL_GETSUB ( ISUBNO, IFNCPTR )
```

Sets IFNCPTR to be a pointer to user subroutine number ISUBNO, according to the order the subroutines are specified in the USER-MODELS sentence.

After calling this function, you would call your user subroutine using DMS_DOCALL as follows:

```
CALL DMS_DOCALL (IFNCPTR, ARG1, ARG2, ARG3, ..., ARG80)
```

Where ARG1, ..., ARG80 are the eighty arguments to be passed to your user subroutine. If your subroutine uses fewer arguments (ten, in the following example), use a dummy local integer variable (IDUM, in the example) for each additional argument up to ARG80:

```
CALL DMS_DOCALL (IFNCPTR, ARG1, ARG2, ARG3, ARG4, ARG5, ARG6, ARG7, ARG8, ARG9, ARG10, IDUM, IDUM, ..., IDUM)
```

Calling your user subroutine through DMS_DOCALL gives you the flexibility of changing the called subroutine without needing to modify the calling subroutine, provided the called subroutines take the same arguments. For instance, if your model has two alternate methods for calculating a particular property, you could create a subroutine which implements each method, then switch between methods by changing the subroutine specified to User2.

Argument List Descriptions for Other Helper Functions

Variable	I/O	Type	Dimension	Description
IARRAY	I	INTEGER	IARRAYLEN	Hollerith array
IARRAYLEN	I	INTEGER	—	Length of IARRAY
LOC	I	INTEGER	—	Location in IARRAY to start converting
CHARSTR	I/O	CHARACTER	*(*)	Character string
ICHARHOL	O	INTEGER	ICHLEN	Hollerith array
ICHLEN	I	INTEGER	—	Length of ICHARHOL (integer words)
INAME	O	INTEGER	at least 40	Buffer, length at least 40 words, for spreadsheet name, as a Hollerith array
NAMELEN	O	INTEGER	—	Actual number of characters in spreadsheet name
NUMSUBS	O	INTEGER	—	Number of user subroutines
ISUBNO	I	INTEGER	—	1-based subroutine index
IFNCPTR	I/O	INTEGER	—	Subroutine pointer

User3 Fortran Models

The unit operation model User3 allows you to interface your own equation-oriented Fortran models with Aspen Plus. A User3 model consists of a number of different procedures which are initiated by calls to the main model subroutine with different values for the argument K. Commonly, the main model subroutine is just a driver which calls other subroutines as needed to perform the steps required. The report can be generated by the same subroutine, or by a separate routine using the same argument list.

All declarations for the User3 model subroutine can be found in the file u3args.inc which is located in the commons directory of the Aspen Plus simulation engine installation. The initial arguments, up to INTSIZE, are identical to the arguments of User2.

Calling Sequence for User3

```
SUBROUTINE USRxxx† (NMATI, SIN, NINFI, SINFI, NMATO, SOUT, NINFO,
                     SINFO, IDSMI, IDSII, IDSMO, IDSIO, NTOT, NSUBS,
                     IDXSUB, ITYPE, NINT, INT, NREAL, REAL, IDS,
                     NPO, NBOPST, NIWORK, IWORK, NWORK, WORK, NSIZE,
                     SIZE, INTSIZE, LDMAT, NWDIR, IWDIR, MODEL1,
                     MODEL2, MODEL3,K, NDE, NAE, NY, NIV, NAV, Y,
                     YMIN, YMAX, YSCALE, YSBND, YSTAT, F, FSCALE,
                     FSTAT, NZ, ISTOR, JSTOR, BJ, X, XEND, NONNEG,
                     NLAB, LABX, LABY, LABF, KDIAG, NPOINT, XVAL,
                     YVAL, NREPV, IREPV, IFAIL, ISTOP, KSTOP,
                     IRESET, KERROR)
```

[†]Subroutine name you entered on the **User3 | Input | Specifications** sheet. Must begin with USR and have a name no longer than 6 characters.

Argument List Descriptions for User3

Variable	I/O [†]	Type	Dimension	Description
NMATI	I	INTEGER	—	Number of inlet material streams
SIN	I/O	REAL*8	NTOT, NMATI	Array of inlet material streams (see Stream Structure and Calculation Sequence)
NINFI	I	INTEGER	—	Number of inlet information streams
SINFI	I/O	REAL*8	NINFI	Vector of inlet information streams (see Stream Structure and Calculation Sequence)
NMATO	I	INTEGER	—	Number of outlet material streams
SOUT	O	REAL*8	NTOT, NMATO	Array of outlet material streams
NINFO	I	INTEGER	—	Number of outlet information streams
SINFO	O	REAL*8	NINFO	Vector of outlet information streams (see Stream Structure and Calculation Sequence)
IDSMI	I	INTEGER	2, NMATI	IDs of inlet material streams as Hollerith strings
IDSII	I	INTEGER	2, NINFI	IDs of inlet information streams as Hollerith strings
IDSMO	I	INTEGER	2, NMATO	IDs of outlet material streams as Hollerith strings
IDSIO	I	INTEGER	2, NINFO	IDs of outlet information streams as Hollerith strings
NTOT	I	INTEGER	—	Length of material streams
NSUBS	I	INTEGER	—	Number of substreams in material streams
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector

Variable	I/O[†]	Type	Dimension	Description
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Model Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Model Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Model Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Model Parameters)
IDS	I	INTEGER	2, 3	Block IDs as Hollerith strings: (* , 1) - Block ID (* , 2) - User model subroutine name (* , 3) - User report subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 2)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector (see Local Work Arrays)
NSIZE	I	INTEGER	—	Length of size results vector
SIZE	O	REAL*8	NSIZE	Real sizing results (see Size)
INTSIZ	O	INTEGER	NSIZE	Integer size parameters (see Size)
LDMAT	I	INTEGER	—	Plex location of the material stream descriptor bead
NWDIR	I	INTEGER	—	Length of work directory (see Local Work Arrays)
IWDIR	W	REAL*8	NWDIR	Work directory (see Local Work Arrays)
MODEL1	I	INTEGER	—	MODEL1 through MODEL3 are external subroutine names supplied on the User3 setup form (as Hollerith strings).
MODEL2	I	INTEGER	—	
MODEL3	I	INTEGER	—	
K	I	INTEGER	—	Type of action expected (see K Codes)
NDE	I/O	INTEGER	—	Number of differential equations
NAE	I/O	INTEGER	—	Number of algebraic equations
NY	I/O	INTEGER	—	Number of variables
NAV	I/O	INTEGER	—	Number of additional variables
Y	I/O	REAL*8	NDE+NY+NIV+NAV	Variable array
YMIN	I/O	REAL*8	NDE+NY+NIV+NAV	Variable lower bound array
YMAX	I/O	REAL*8	NDE+NY+NIV+NAV	Variable upper bound array
YSCALE	I/O	REAL*8	NDE+NY+NIV+NAV	Variable scaling array
YSBND	I/O	REAL*8	NDE+NY+NIV+NAV	Variable step bound array
YSTAT	O	INTEGER	NDE+NY+NIV+NAV	Variable status array

Variable	I/O [†]	Type	Dimension	Description
F	O	REAL*8	NDE+NAE+NIV	Function residual array
FSCALE	I/O	REAL*8	NDE+NAE+NIV	Function residual scaling factor array
FSTAT	O	INTEGER	NDE+NAE+NIV	Function status array
NZ	I/O	INTEGER	—	Number of Jacobian non-zero elements
ISTOR	I/O	INTEGER	NZ	Row (functions) indices of Jacobian non-zeroes (see Sparsity Example)
JSTOR	I/O	INTEGER	NZ	Column (variable) indices of Jacobian non-zeroes (see Sparsity Example)
BJ	O	REAL*8	NZ	Jacobian value array
X	I/O	REAL*8	—	DAE Integration variable
XEND	O	REAL*8	—	Limit of DAE integration variable
NONNEG	O	INTEGER	NDE+NY+NIV+NAV	Non-negative flag
NLAB	I/O	INTEGER	—	Length of names
LABX	I/O	INTEGER	NLAB, *	Integration variable name array
LABY	I/O	INTEGER	NLAB, NDE+NY+NIV+NAV	Variable name array
LABF	I/O	INTEGER	NLAB, NDE+NAE+NIV	Function names array
KDIAG	I	INTEGER	—	History output level (SIM-LEVEL)
NPOINT	I	INTEGER	—	Number of points for history tabulation
XVAL	I	REAL*8	NPOINT	Array of integration values at tabulation points
YVAL	I	REAL*8	NDE+NY+NIV+NAV, NPOINT	Array of variable values at tabulation points
NREPV	—	INTEGER	—	Not used
IREPV	—	INTEGER	NREPV	Not used
IFAIL	O	INTEGER	—	Model evaluation failure flag (1 = cannot evaluate functions)
ISTOP	I	INTEGER	—	Index for variable which reached a stop criterion (DAE)
KSTOP	I	INTEGER	—	Stop reason code (DAE) (1 = max value reached, -1 = min value reached)
IRESET	O	INTEGER	—	Reset flag (1 = partial reset, 2 = total reset, 3 = total reset for new problem)
KERROR	O	INTEGER	—	Error flag (1 = input checking error, negative = integrator error)

[†]I = Input to subroutine, O = Output from subroutine, W = Workspace

Stream Structure and Calculation Sequence

The stream structure for material streams is described in Appendix C. For information streams, the stream vector consists of a single value. All stream data are in SI units.

In normal sequential modular calculations, the block calculates (fills in) the outlet streams based on the inlet streams and specifications you made for the

block. However, sometimes it is useful for a block to modify its feed stream(s), such as when the feed stream has no source block because it is a feed to the process. To handle these cases, the user subroutine may change its inlet stream.

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed if property calculations are to be performed using the first (or only) option set. NBOPST(1,2) should be passed if property calculations are to be performed using the second option set.

USER3 Data Classifications

A USER3 model handles several types of data: control information, model parameter data, open model variables, and local work arrays.

USER3 Control Information

Control information is defined by the USER3 system and passed between the USER3 executive and the user-written subroutines. This includes various control flags and system parameters which the user-written model is expected to either interpret or set. An example is the K flag, which indicates the type of information the USER3 executive expects from the user-written model on any given call.

Model Parameters

Parameters are those values which do not change during a run, thus they are not model variables. Model parameters are read from either the REAL or the INT array. An example of an integer parameter is the number of stages in a tower or the number of product streams leaving the block. Examples of real parameters are values of regression coefficients. Integer parameters may be used to dimension arrays, determine the number of equations necessary, or point to array locations, while real parameters are typically used as constants in equations.

To use these parameters, specify the number of them on the User3 Input User Arrays sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters, and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

USER3 Variable Data Structures

Each variable has several attributes, such as value, upper bound, lower bound, step bounds, status, scaling factor and name. All model variable values associated with a given block (including the attached stream variables) are stored in the Y array. Other variable attributes are stored in similarly

dimensioned arrays, such as YMAX, YMIN, YSBND, YSTAT, YSCALE, and LABY corresponding to the attributes listed above.

Variable values can appear in several contexts within USER3. The first is within the Plex, which is the Aspen Plus linked list data structure. Within the Plex, stream variables are stored in stream beads (a bead is an area of contiguous memory locations) and USER3 block variables are kept in the Y array corresponding to the Y keyword input sentence. REAL and INT array parameters for a given block are stored in an assigned portion of the Plex. A second context might be as a local name, facilitating the writing of easily understood equations with meaningful variable names. A third context is the global vector which is passed between the model and the internal or external solver. These contexts are given the shorthand designations P for Plex, V for Local and X for Global. In many cases it is necessary to transfer values from one context to another, as in Plex to Global (PTX) or Plex to Local (PTV) and Local to Global (VTX) during initialization.

The Y variable vector is passed to the USER3 model via the calling argument list. On input, depending on the value of the USER_IRESTR flag in COMMON /PPEXEC_USER/, the Y vector may either be the initial Y array set in the keyword input file (USER_IRESTR=1), or a persistent Y vector which contains the latest updated Y values (USER_IRESTR=2). On output, the Y vector contains the values of all variables associated the block.

The model writer is responsible for providing the local names for the block variables generated by the USER3 model. The fully qualified name is formed by prepending the block id and the string '.BLK' to the name. The names of the stream variables are automatically provided by the stream utilities which generate the stream variables.

Local Work Arrays

You can use local work arrays by specifying the array lengths on the User3 Input Specifications sheet. Aspen Plus does not retain these arrays from one call to the next.

Size

The size array SIZE should be filled in with the results from simulation calculations if sizing and costing calculations are performed. The values that are stored depend on the cost model. The values can be stored in any order. For each element in SIZE, the corresponding element in INTSIZ should be filled out with the correct result code. The result codes are:

- | | |
|----|---|
| 10 | Inlet pressure (N/m ²) |
| 11 | Outlet pressure (N/m ²) |
| 12 | Inlet temperature (K) |
| 13 | Outlet temperature (K) |
| 14 | Inlet vapor volumetric flow (m ³ /s) |
| 15 | Outlet vapor volumetric flow (m ³ /s) |
| 16 | Outlet liquid volumetric flow (m ³ /s) |

17	Vapor density (kg/m ³)
18	Liquid density (kg/m ³)
19	Heat duty (watt)
50	Area (m ²)
51	Isentropic or polytropic efficiency
52	Cp/Cv
53	Mechanical efficiency
54	Compressor type (1=polytropic centrifugal; 2=polytropic positive displacement; 3=isentropic)
55	Horsepower (watt)
92	Surface tension (N/m)
93	Liquid viscosity (N-s/m ²)
94	Relative volatility
95	Diffusivity (m ² /s)

Variable Types and Mapping Concepts

USER3 models work primarily with two types of open variables: stream variables and block variables. Mole flow based material stream variables are:

blkid.strid.STR.compид.FLOW	Component Molar Flows
blkid.strid.STR.FLOW	Total Molar Flow
blkid.strid.STR.ENTH	Enthalpy
blkid.strid.STR.PRES	Pressure

Information stream variables are:

blkid.strid.STR.HEAT	Heat stream
blkid.strid.STR.WORK	Work stream

Block variables have names of the form:

blkid.BLK.lcid

Stream variables may be created by calling stream utilities or directly by the model. Block variables are created by the model or by open physical property calls.

Variables may be accessed for residual and Jacobian computation in several ways. The preferred method is to write the equations using the Y array and indices to designate the variable locations. For example:

```
F(1) = Y(KPO(1)) - Y(KPI(1)) - Y(KDELP)
```

An alternate method is to write the equations in terms of local variable names and explicitly map the values between the global Y array and the local names. For example:

```
POUT = Y(KPO(1))
```

```

PIN = Y(KPI(1))
DELTAP = Y(KDELP)
F(1) = POUT - PIN - DELTAP

```

In the preferred method, the Y array locations are mapped symbolically by the VMP routine. This allows all operations on Y and Y attribute arrays to be defined in terms of symbolic indices which do not change when their Y location changes. For example:

VMP routine fragment:

```

N = N + 1
KQBYN = N
N = N + 1
KQACT = N
N = N + 1
KSPEED = N

```

RES routine fragment:

```

M = M + 1
F(M) = Y(KSPEED) * Y(KQBYN) - Y(KQACT)

```

Note that creating a new variable changes only the VMP routine, since the symbolic pointer names for existing variables do not change when the numerical index changes. Thus, existing residuals in the RES routine remain unchanged.

Stream variables are mapped by the U3SVMP routine as follows:

KFI(1..NMATI, 1..NCF)	Component flows of each inlet material stream
KFTI(1..NMATI)	Total flow of each inlet material stream
KHI(1..NMATI)	Molar enthalpy of each inlet material stream
KPI(1..NMATI)	Pressure of each inlet material stream
KFINFI(1..NINFI)	Energy flow of each inlet information stream
KFO(1..NMATI, 1..NCF)	Component flows of each outlet material stream
KFTO(1..NMATI)	Total flow of each outlet material stream
KHO(1..NMATI)	Molar enthalpy of each outlet material stream
KPO(1..NMATI)	Pressure of each outlet material stream
KFINFO(1..NINFO)	Energy flow of each outlet information stream

Block variables are mapped by a user-written routine USRxxx_VMP. By convention, mapping indices begin with the letter 'K'. Arrays of block variable mapping indices must be locally dimensioned. For example:

```

INTEGER*4 MXCF, MXSOUT
PARAMETER (MXCF = 50, MXSOUT = 20)
INTEGER*4 KDELP, KALPHA(MXSOUT, MXCF)

```

To reduce the lengths of internal model argument lists, a model-specific include file is created with a common block to hold the block and stream variable mapping indices. This include file is typically called USRxxx_vmp.INC and is included in the USER3 model main subroutine and all model subroutines which access variables or attributes. For example, an insert file called 'usrtoa_vmp.inc' might consist of:

```

integer*4 mxmati, mxmato, maxinf, mxcf
parameter (mxmati=1, mxmato=1, maxinf=1, mxcf=50)
integer*4 kfti, kfi, khi, kpi, kfinfi, kfto, kfo, kho, kpo,

```

```

kfinfo
  INTEGER*4 KTEMP, KCP, KMOLV, KMW, KQACT
  common /usrtoa_vmp/ kfti(mxmati), kfi(mxcf,mxmati), khi(mxmati),
& kpi(mxmati), kfinfi(maxinf), kfto(mxmato), kfo(mxcf,mxmato),
& kho(mxmato), kpo(mxmato), kfinfo(maxinf),
& KTEMP, KCP, KMOLV, KMW, KQACT

```

Template USER3 model-specific include files are distributed with Aspen Plus in the Engine\Custom directory.

Scaling and Units Conversion in USER3

Aspen Plus uses group scaling factors for enthalpies, flows, temperatures, K-values, and pressures. These are accessed by calling the utility U3GSCL. The model writer should provide a scaling factor for each variable and function and place these in arrays YSCALE and FSCALE. By default those scaling values which are not provided by the user are assigned a value of 1.

In order to have consistency across connection equations within Aspen Plus, all stream variables should be scaled by the appropriate group scaling factors. Any block variables for which a group scaling factor exists should also be scaled by that value.

Currently a USER3 model must be internally consistent with SI units. The model calculations may use any set of units as long as all data are converted from SI upon entering the model and to SI upon exiting the model. Two units conversion functions are provided: IU_UU2SI(VALUE, NROW, UNITS) and IU_SI2UU(VALUE, NROW, UNITS), where VALUE is the real value to be converted, NROW is the unit group, and UNITS is a valid Aspen Plus unit name from the units of measure table in *Aspen Plus Input Language*, Chapter 3. NROW is generally unnecessary except to distinguish temperature (22) from temperature change (31).

Variables passed to the model via the SPECS or Y sentences are not converted or modified in any way. Thus, it is up to the model writer to specify the units sense of input for the model user. All Aspen Plus USER3 library models expect input in SI units. The VARIABLES sentence will convert values from the stated *uom* to SI units.

K Codes

The USER3 model returns variable, equation, and Jacobian array dimensions; names the variables and equations; provides a cold or warm initialization; returns residual or Jacobian values; writes to the history or report files; or transfers variable values back into the Plex storage structure depending on the value of flag K passed to the model. Each type of request is detailed in subsequent paragraphs. The values of K and corresponding actions are:

K	Action
0	Cold initialization
1	Compute residual values
6	Presolve
7	Solver termination: Store solution in the Plex

K	Action
9	Compute Jacobian values
10	Warm initialization
13	Return array dimensions
14	Return variable names
15	Return model fixed/free status changes
101	Write to report file

Array Dimensions (K=13)

At this call the model returns the total number of variables (NY), the total number of equations (NAE) and the total number of nonzero Jacobian elements (NZ). If the model processes material and/or information streams and you want to use the standard material and energy balances, there are stream utility calls which return dimensions for the stream variables and equations. The model must then add the values required for its block variables, equations and nonzeros to those from the stream utilities before returning.

Variable and Function Names (K=14)

This call requests the model to return the list of variable names in the array LABY and function names in the array LABF. The variable names are processed by the USER3 interface routine in order to match the indices to the names given in the SPECS, VARIABLES or FUNCTIONS sentence. Both sets of names are passed to the solver and used for reporting purposes.

Variable Fixed/Free Status (K=15)

This call allows the model to change the variable fixed/free status. This may be useful under certain circumstances, although in most cases no action is necessary at this call.

Cold Initialization (K=0)

At this step the model performs several functions. It copies the data from the Plex to Local and/or Global variables. Data from the Plex consists of material and information streams, initial or persistent Y variables and values entered in the REAL and INT array for the block being modeled. Local variables are those with convenient local names for use in writing the model equations, while global variables are those passed in the Y variable vector as used by the internal or external solver. During this operation you can initialize any variables whose values are not given or unavailable, for example those which the user does not initialize in the user interface and which therefore appear as the value RMISS. Variable and equation scaling factors and variable upper, lower and step bounds should be determined and loaded into the YSCALE, FSCALE, YMAX, YMIN, and YSBND vectors respectively. The sparsity pattern for the Jacobian matrix is loaded into the ISTOR and JSTOR arrays for the row and column indices for each nonzero element.

Warm Initialization (K=10)

This generally includes a small subset of the cold initialization activities. The primary purpose of this call is to give the model an opportunity to take action before solution starts, for example, to read a local data file or set up non-retained run-time pointers.

Presolve (K=6)

At this call the model can make a final presolve error check or perform any desired presolve actions.

Return Residual Values (K=1)

At this call, the solver passes in the current values of the variables and the user model is responsible for returning values for each residual equation.

Return Jacobian Values (K=9)

At this call the current values of the variables are passed in by the solver and the user model is responsible for returning values for each Jacobian element.

Write to the Report File (K=101)

During this call, which occurs after flowsheet solution, the model writes results to the report file. The FORTRAN logical unit number for the report file is given by the value of USER_NRPT from the PPEXEC_USER common block. Note that USER3 provides information for all streams connected to USER3 blocks, therefore the model writer need only print non-stream information and perhaps selected stream variables in the block section of the report file.

Solver Termination: Copy Values into the Plex (K=7)

At this call, the model copies all values from Global or Local to Plex. By default, USER3 creates an equation bead to provide persistent storage of the equation-oriented data, for example the Y and F arrays, Y and F attribute arrays, and Jacobian pattern and value arrays. Typically, at this message the model needs only to call U3SPTX to store stream values into the Aspen Plus stream bead.

Sparsity Example

A simple model to estimate pressure drop across a unit consists of 5 variables and 2 equations:

$$\begin{aligned}f(1) &= \text{DELTAP} - \text{PRES_PARAM} * \text{MASS_FLOW}^2 = 0 \\f(2) &= \text{PRES_IN} - \text{PRES_OUT} - \text{DELTAP} = 0\end{aligned}$$

Assume the variables are ordered and named as:


```

Y(1) = DELTAP
Y(2) = PRES_IN
Y(3) = PRES_OUT
Y(4) = PRES_PARAM
Y(5) = MASS_FLOW

```

and the functions are named as:

```

f(1) = ESTIMATE_DELTAP
f(2) = DELTAP_DEFINITION

```

The sparsity pattern (rows=equations; columns=variables) is:

	1	2	3	4	5
1	x			x	x
2	x	x	x		

The Jacobian matrix of first partial derivatives is:

```

df(1)/dy(1) = 1
df(1)/dy(4) = -MASS_FLOW^2
df(1)/dy(5) = -2 * PRES_PARAM * MASS_FLOW
df(2)/dy(1) = -1
df(2)/dy(2) = 1
df(2)/dy(3) = -1

```

This information would be returned by the model as follows:

On the array dimensions (K=13) call:

Set NZ = 6.

On the cold initialization (K=0) call:

Set ISTORE = 1, 1, 1, 2, 2, 2.

Set JSTORE = 1, 4, 5, 1, 2, 3.

On the return Jacobian values (K=9) call:

Set BJ = 1, -MASS_FLOW^2, -2 * PRES_PARAM * MASS_FLOW, -1, 1, -1
(using the current values of MASS_FLOW and PRES_PARAM).

To solve the system in the square case, 3 variables must be specified. One valid specification is to fix PRES_IN, PRES_OUT, and MASS_FLOW to compute PRES_PARAM and DELTAP. Another is to fix PRES_IN, MASS_FLOW, and PRES_PARAM to compute DELTAP and PRES_OUT.

Creating a USER3 Model

The standard methodology for writing a USER3 model may be summarized as:

- 1 Develop the model on paper.
- 2 Design the INT and REAL arrays.
- 3 Decide which stream utilities and open physical property calls apply.
- 4 Determine initialization procedure.
- 5 Fill in the template model main subroutine.
- 6 Create the model mapping include file.
- 7 Design the output format.

- 8 Fill in the template subroutines.
- 9 Compile, debug and test on Aspen Plus side.

Step 1: Develop the Model on Paper

- Write out variable names.
- Write out equations in residual format.
- Name the equations.
- Write out the sparsity pattern.
- Write out Jacobian values.

Step 2: Design the INT and REAL Arrays

- The standard INT convention is to have:
INT(1) = NDHEL (Number of Directory Header Elements - 3)
INT(2) = NINTP (Number of Integer Parameters)
INT(3) = NREALP (Number of Real Parameters)
INT(4) = IMODE (Sequential Mode Flag)
- Determine necessary integer parameters and their order in the INT array.
- Decide if additional INT storage is necessary for model internal use.
- Set length of INT array, NINT.
- Determine necessary real parameters and their order in the REAL array.
- Decide if additional REAL storage is necessary for model internal use.
- Set length of REAL array, NREAL.

Step 3: Decide Which Stream Utilities & Open Physical Property Calls Apply

- Determine whether overall energy balance, overall material balance, or component balances are required.
- Determine which, if any, open physical property calls are necessary.

Step 4: Determine Initialization Procedure

- All uninitialized variables should be given reasonable values.
- Initialization can be simply to provide default values or to perform sequential computations based on information given by the user.

Step 5: Fill in the Template Model Main Subroutine

- Modify template calls to reflect the name of the model and its subroutines.
- Modify stream utility and physical property calls as appropriate.
- Customize main subroutine as required.

Step 6: Create Model Mapping Include File

- Determine maximum dimensions for number of streams and components.
- Dimension local arrays.
- Declare stream and block variables and place into COMMON /USRxxx_VMP/.

Step 7: Design the Output Format

- Decide what block report should look like.
- Decide if additional information (beyond what is printed by the USER3 executive) is necessary.

Step 8: Fill in Template Subroutines

- Rename each template subroutine.
- Include the block include file.
- For each subroutine, insert block specific code.
- Customize open physical property wrapper subroutines for the specific block.
- Write the block specific report routine.
- Write the block specific initialization routine.

Step 9: Compile, Debug, Test in Aspen Plus

- Compile: ASPCOMP USRxxx.FOR in an Aspen Plus Simulation Engine window.
- Run Aspen Plus.
- Use interactive kernel or calls to debugging utility subroutine to debug problems.
- Run several versions of the test problem with different specifications, input values, etc.

Additional User3 Subroutines

USER3 consists of a number of distinct collections of subroutines. These include the software which provides the interface to the Aspen Plus routines and data structures, the internal and external solvers, the utilities directly available to the model writer for manipulating data, character strings, process and information streams, and the user-written model subroutines themselves. The calling argument list and functional description of the USER3 model-writing utilities are given below.

The main USER3 subroutine must begin with USR and have 6 or fewer characters and is represented in the names below as USRxxx. All other user-written subroutines must begin with the letters USR, but have no restriction on name length. Those utilities which are generic to all USER3 models are named U3xxxx and those which manipulate stream variables and functions are named U3Sxxx. For a USER3 model processing material and/or information streams the following routines are typically needed:

- USRxxx - Main model subroutine.
- U3SVMP - Maps stream variable locations (index in Global variable vector).
- USRxxx_VMP - Maps block variable locations (index in Global variable vector).

K = 13 (Return Dimensions):

- U3HDR - Reads the INT array header.
- USRxxx_IP - Reads integer parameters from INT array.
- USRxxx_RP - Reads real parameters from REAL array.
- USRxxx_CHK - Checks data input from block paragraph.
- U3_MSTRM_NCF - Computes the number of components in a material stream (full slate).
- U3_MSTRM_NZC - Computes actual number of non-zero components in a material stream.
- U3SNY - Returns the number of stream variables.
- U3SNF - Returns the number of stream residuals.
- U3SNNZ - Returns the number of stream non-zero Jacobian elements.
- USRxxx_DIM - Returns the number of block variables, residuals, and Jacobian elements.

K = 14 (Return Names):

- U3SVL - Assigns names to the stream variables.
- U3SEL - Assigns names to the stream functions.
- USRxxx_NAM - Assigns names to the block variables and functions.
- U3DNAM - Assigns default names to other vars. & eqns.

K = 0 (Cold Initialization):

- U3GSCL - Computes group scaling factors.
- U3SCPY - Initializes outlet stream vector.
- U3SPTX - Copies stream variables from Plex -> Global.
- USRxxx_PTX - Assigns attributes to Global block variables.
- USRxxx_INI - Custom closed-form initialization.
- U3VPPT - Put variable physical type.
- U3SETF - Packs stream equations into Global vector.
- USRxxx ETF - Packs block equations into Global vector.
- U3SPAT - Generates sparsity pattern for stream equations.
- USRxxx_PAT - Generates sparsity pattern (and nz) for Jacobian of block equations.

K = 10 (Warm Initialization):

No default routines.

K = 6 (Presolve):

No default routines.

K = 1 (Residual Evaluation):

- U3SRES - Evaluates stream equation residuals.
- USRxxx_RES - Evaluates block equation residuals.

K = 9 (Jacobian Evaluation):

- U3SJAC - Evaluates stream equation Jacobian elements.
- USRxxx_JAC - Evaluates block equation Jacobian elements.

K = 7 (Call After Solver Termination):

- U3SOTP - Copies stream variables from Global -> Plex.
- USRxxx_OTP - Copies block variables from Global -> Plex.

K = 101 (Write to Report):

- USRxxx_RPT - Writes to the Report file.

If open physical property calls are used, they are typically collected into a set of wrapper calls with parallel functionality to the stream and block variable calls. A set of template physical property call wrappers, illustrating how each type of open physical property call is used, is included in the **Aspen Plus** `<version>\Engine\User` folder as file PPTMP.FOR.

These user-written physical property call wrapper routines are typically named:

- USRxxx_PP_DIM - Count additional variables, functions and nonzeros created by open calls.
- USRxxx_PP_NAM - Name the additional variables and functions created by the open calls.
- USRxxx_PP_PAT - Provide sparsity pattern for Jacobian rows added by the open calls.
- USRxxx_PP_ETF - Scale the functions generated by the open calls.
- USRxxx_PP_RES - Return residual values of the open call functions.
- USRxxx_PP_JAC - Return the Jacobian values for the elements added by the open calls.

In general, the open calls only create variables which are internal to that call. For example, temperature, pressure, or component flow variables used in enthalpy, molar volume, or heat capacity calls are not created, since those are typical stream or block variables. In some cases, notably the Open Two-Phase Flash Interface (O2FI), many variables are created automatically (for example K-values) and many others may be created by either the user or the O2FI call at the user's discretion (for example, temperature or duty may or may not already exist as block variables).

The stream utilities, block routines and open physical property call wrappers normally have the same extension, but where different the stream utility routine name endings are given in parentheses.

Input Processing: U3HDR, IP, RP, CHK Subroutines

Input processing routines retrieve and check data from the REAL and INT arrays. U3HDR reads the standard RT-Opt INT array header:

INT(1) = NDHEL Number of directory header elements (3)
INT(2) = NINTP Number of (block specific) integer parameters
INT(3) = NREALP Number of real parameters (for input checking)

(It is also customary to set INT(4) = IMODE, the sequential mode flag, but this is not read by U3HDR.) USRxxx_IP reads block specific integer parameters from the INT array and assigns them to local names. USRxxx_RP reads the REAL array and assigns the values to local names. USRxxx_CHK performs block specific input data checking.

Mapping: VMP Subroutine

Each time the USER3 model is called, the VMP subroutine is invoked to map variable Y locations to symbolic index names. The symbolic indices are stored in a common which is included in all subroutines which access variable arrays.

Dimensions: DIM (or NY, NF) Subroutines

At message K=13 the model is requested to provide the number of variables, functions and nonzero Jacobian elements. Stream utility subroutines provide this information for the selected types of stream variables and functions. The block specific USRxxx_DIM routine provides this information for the block variables and functions and increments the corresponding dimensions. Open physical property calls, if used, also increment the dimension counters.

Naming: NAM (or VL, EL) Subroutines

At message K=14 the model is requested to provide the names of functions and variables. Stream utility subroutines provide names for the selected types of stream variables and functions. The block specific USRxxx_NAM routine provides names for the block variables and functions. If open physical property calls are used, USRxxx_PP_NAM provides names for any additional variables and functions thus created.

Initialization: INI, PTX, ETF Subroutines

At message K=0 the model is requested to initialize itself. Initialization activities include providing values, bounds, physical type and scale factors for variables and scale factors for functions. Stream utility subroutines perform these activities for the selected types of stream variables and functions. The block specific USRxxx_NAM routine provides names for the block variables and functions. If open physical property calls are used, USRxxx_PP ETF scales those functions.

Pattern: PAT Subroutine

At message K=0 the model is requested to provide the Jacobian pattern. Stream utility subroutines perform these activities for the selected types of stream functions. The block specific USRxxx_PAT routine provides pattern and nz dimension for the block functions. If open physical property calls are used, USRxxx_PP_PAT provides the sparsity pattern and nz for those functions.

Function Evaluation: RES Subroutine

At message K=1 the model is requested to return its function residual values. Stream utility subroutines do this for the selected types of stream functions. The block specific USRxxx_RES routine returns values of the block functions. When open physical property calls are used, USRxxx_PP_RES returns their function values.

Output: RPT Subroutine

At message K=101 the model is called to write to the Report file. Block specific routine USRxxx_RPT is prepared to create customized output. This subroutine can access the FORTRAN logical unit number for the Report file from COMMON /PPEXEC_USER/. This subroutine can call several Aspen Plus user interface output utilities.

Closure: XTP Subroutine

At message K=7 the model is called upon solution or termination. The USER3 model can call stream utility routines to copy stream values into the Aspen Plus stream arrays. The model can also take other block specific actions as necessary.

Jacobian Evaluation: JAC Subroutine

At message K=9 the model is called to evaluate the Jacobian. Stream utility subroutines do this for the selected types of stream functions. The block specific USRxxx_JAC routine returns block Jacobian values. If open physical property calls are used, USRxxx_PP_JAC returns their Jacobian values. If the numeric layer is used the model will not be called with this message level.

USER3 Stream Utilities

Stream utilities (U3Sxxx) create standard RT-Opt stream variables (mole flow basis) for each material and information stream present and writes standard material and energy balances. The stream functions are:

- Overall energy balance.
- Overall material balance.
- Component material balances.

Other stream utilities are available for writing only subsets of these functions, such as energy balance only, overall material balance only, or balances on a subset of the components.

For variables, the U3Sxxx routines are:

- NY Number of variables.
- VL Variable names.
- VMP Variable mapping.
- PTX Variable and attribute initialization.

For the full set of material and energy balance functions and their attributes, the U3Sxxx routines are:

- NF Number of functions.
- NZ Number of nonzero Jacobian elements.
- RES Function residual evaluation.
- PAT Jacobian sparsity pattern.
- JAC Jacobian evaluation.
- EL Function names.
- ETF Function scaling factors.

Subroutine U3SCPY copies one Aspen Plus stream array into another, typically to aid in initialization of outlet streams. Subroutine U3SPTX copies values of all Aspen Plus streams associated with the block into the appropriate stream variables and also initializes the stream variable attributes (scales and bounds). Subroutine U3SXTP copies values of all stream variables into the proper Aspen Plus streams and performs a flash calculation to fill in other stream properties.

The USER3 stream utilities work internally with stream descriptors, which contain information about the stream name, location, and number of components. The USER3 model must call U3SDESC or U3SDESC1 to initialize the stream descriptors if stream utilities are used. U3SDESC1 supports the use of a non-zero component group, while U3SDESC does not. If U3SDESC1 is used, the user-specified non-zero component list may be obtained with subroutine U3GIDX_BLK. Parameters for sizing the stream descriptors are declared and stored in include files 'U3MSTRM.INC' and 'U3ISTRM.INC'.

Material Balance Only Routines

If the USER3 model requires no energy balance or a custom energy balance, stream utilities which only write the material balance functions may be called. These routines work with overall and component material balances and have names of the form U3_MBAL_xxx, where xxx represents the standard USER3 subroutine nomenclature:

- NF Number of functions.
- NZ Number of nonzero Jacobian elements.
- RES Function residual evaluation.
- PAT Jacobian sparsity pattern.
- JAC Jacobian evaluation.
- NAM Function names.
- SCL Function scaling factors.

The material balance stream utilities can write functions on selected components only - for example, inerts in a reacting system. The number of

components for which an overall material balance is requested is indicated by argument NINDX. The components to be balanced are indicated in the array INDX(NINDX), by their location in the Aspen Plus stream slate. The components appear in the Aspen Plus stream slate in the same order they are defined in the input language. The index of a component is obtained by calling subroutine U3_GET_COMPIINDEX(NAME, IDX).

Energy Balance Only Routines

If the USER3 model requires no material balance or custom material balances, stream utilities which only write the energy balance function may be called. These routines have names of the form U3_EBAL_xxx, where xxx represents the standard USER3 subroutine nomenclature, just as in the case with the U3_MBAL_xxx routines.

Energy Balance Work and Heat Sign Conventions

The USER3 energy balance assigns a positive value to energy entering via material flow and a negative value to energy leaving via material flow. Inlet information streams are positive when energy enters the block and negative when energy leaves the block. Outlet information streams are positive when energy leaves the block and negative when energy enters the block. Information streams are either HEAT or WORK. Inlet HEAT streams are added to the energy balance and outlet HEAT streams are subtracted from the energy balance. Inlet WORK streams are subtracted from the energy balance and outlet WORK streams are added to the energy balance. The energy balance is of the form:

$$\sum_{i=\text{nmati}} \text{FTI}(i) * \text{HI}(i) - \sum_{i=\text{nmato}} \text{FTO}(i) * \text{HO}(i) \pm \sum_{i=\text{ninfo}} \text{FINFI}(i) \pm \sum_{i=\text{ninfo}} \text{FINFO}(i) = 0$$

Open Physical Property Calls

Open physical property calls provide a mechanism for a USER3 model to incorporate modular subroutines for determining physical properties within the model. These modules are themselves open models and may be appended or imbedded within a larger open model. There are currently 5 single-phase open properties available. Some of these have both mole flow and mole fraction based subroutine calls. The properties are:

- Single-phase mixture enthalpy (mole flow and mole fraction).
- Mixture molecular weight (mole flow only).
- Single-phase mixture molar volume (mole flow only).
- Single-phase mixture heat capacity (mole flow only).
- Single-phase mixture entropy (mole flow and mole fraction).

There is also a set of Open Two-Phase Flash Interface (O2FI) calls available for setting up an open vapor-liquid equilibrium calculation for any material "stream". This interface is described fully in the section *Open Two-Phase Flash Interface*. Note that there are two formulations available – a mole flow based formulation and a mole fraction based formulation with extended vapor fraction (PML formulation).

It is convenient to wrap the physical property calls for a specific block into a set of subroutines named USRxxx_PP_xxx, where xxx represents:

- DIM Number of functions and variables.
- RES Function residual evaluation.
- PAT Jacobian sparsity pattern.
- JAC Jacobian evaluation.
- NAM Function and variable names.
- ETF Function scaling factors.

Once the wrappers have been written for a particular block, they are called in the same section of code as the corresponding stream and block functions. A template set of wrapper routines is distributed as file PPTMP.FOR and is easily customized for a particular block.

Differences Between Open & Closed Physical Property Calls

All USER2 closed physical property calls are available within USER3, but only a subset of these calls have been converted to open subroutine modules. Closed property calls return the value of the property when given the proper inputs, while open physical property calls perform all functions required of an open model, including:

- Creating and naming any additional variables.
- Writing a function or functions for the property in residual format.
- Providing Jacobian sparsity pattern and values.

Each single-phase open property call creates no new variables and one new function. The calls for mole flow streams can be made either through the SET routine for the property (by setting the ICFLAG variable for the desired operation), or by individually calling the lower level routines for dimensions, names, sparsity, residuals and Jacobian. The mole fraction stream utilities must be called via the low-level routines since no SET routines exist for those. Note that the mole fraction based calls use the same name subroutine as the mole flow calls, since the name of the open function is the same in each case.

The names and argument list for the high-level SET routines are given first, followed by the description of the argument variables. The low-level subroutines use subsets of the SET arguments. Their names and argument lists follow the variable descriptions.

Physical Property Call Primitives

Single phase heat capacity

```
SUBROUTINE RTO_CPSET      (NCP,KPH,NBOPST,IDX,FLOW,PRES,TEMP,CP,
1                          JXFLOW,JXPRES,JXTE,JXCP,TOLER,FSCALE,
2                          PSCALE,TSCALE,LODIAG,NLAB,ICFLAG,IRNDX,
3                          JXNDX,NZNDX,ZFLOW,ICNAME,XNAME,RNAME,
4                         IRSTOR,JXSTOR,RESID,DRESDX)
```

Single phase enthalpy

```
SUBROUTINE RTO_ENSET      (NCP,KPH,NBOPST,IDX,FLOW,PRES,TEMP,ENTH,
1                          JXFLOW,JXPRES,JXTEMP,JXENTH,TOLER,FSCALE,
2                          PSCALE,LODIAG,NLAB,ICFLAG,IRNDX,JXNDX,
3                          NZNDX,ZFLOW,ICNAME,XNAME,RNAME,IRSTOR,
4                          JXSTOR,RESID,DRESDX)
```

Average molecular weight

```
SUBROUTINE RTO_MWSET      (NCP,IDX,FLOW,AMW,JXFLOW,JXAMW,NLAB,
1                          ICFLAG,IRNDX,JXNDX,NZNDX,ZFLOW,ICNAME,
2                          XNAME,RNAME,IRSTOR,JXSTOR,RESID,DRESDX)
```

Single phase molar volume

```
SUBROUTINE RTO_VMSET      (NCP,KPH,NBOPST,IDX,FLOW,PRES,TEMP,VM,
1                          JXFLOW,JXPRES,JXTEMP,JXVM,TOLER,FSCALE,
2                          PSCALE,LODIAG,NLAB,ICFLAG,IRNDX,JXNDX,
3                          NZNDX,ZFLOW,ICNAME,XNAME,RNAME,IRSTOR,
4                          JXSTOR,RESID,DRESDX)
```

Single phase entropy

```
SUBROUTINE RTO_SMSET      (NCP,KPH,NBOPST,IDX,FLOW,PRES,TEMP,ENTR,
1                          JXFLOW,JXPRES,JXTEMP,JXENTR,TOLER,FSCALE,
2                          PSCALE,LODIAG,NLAB,ICFLAG,IRNDX,JXNDX,
3                          NZNDX,ZFLOW,ICNAME,XNAME,RNAME,IRSTOR,
4                          JXSTOR,RESID,DRESDX)
```

Argument List for Physical Property Call Primitives

Variable	ICFLAGS	Type	Dimension	Description
NCP	I: 0, 1, 2, 3, 4	I	1	Number of packed components.
KPH	I: 1, 3, 4	I	1	Phase code. 1 - Vapor. 2 - Liquid.
NBOPST	I: 3, 4	I	6	Physical property option set.
IDX	I: 1, 3, 4	I	NCP	Non-zero component index.
FLOW	I: 3, 4	R	NCP	Component flows.
PRES	I: 3, 4	R	1	Pressure.
TEMP	I: 3, 4	R	1	Temperature.
CP	I: 3, 4	R	1	Single phase heat capacity variable.
ENTH	I: 3, 4	R	1	Single phase enthalpy variable.
AMW	I: 3, 4	R	1	Average molecular weight variable.
VM	I: 3, 4	R	1	Single phase molar volume variable.
ENTR	I: 3, 4	R	1	Single phase entropy variable.
JXFLOW	I: 2	I	1	Variable vector component flows index.
JXPRES	I: 2	I	1	Variable vector pressure index.
JXTEMP	I: 2	I	1	Variable vector temperature index.
JXCP	I: 2	I	1	Variable vector heat capacity index.

Variable	ICFLAGS	Type	Dimension	Description
JXENTH	I: 2	I	1	Variable vector enthalpy index.
JXAMW	I: 2	I	1	Variable vector molecular weight index.
JXVM	I: 2	I	1	Variable vector molar volume index.
JXENTR	I: 2	I	1	Variable vector entropy index.
TOLER	I: 4	R	1	Residual convergence tolerance.
FSCALE	I: 4	R	1	Flow scale.
PSCALE	I: 4	R	1	Pressure scale.
TSCALE	I: 4	R	1	Temperature scale.
LODIAG	I: 3, 4	I	1	Physical property diagnostic level.
NLAB	I: 1	I	1	Length of variable or residual names.
ICFLAG	I: 0, 1, 2, 3, 4	I	1	Calculation flag. 0 - Return dimensions. 1 - Return variable and equation names. 2 - Return jacobian sparsity. 3 - Return function residual. 4 - Return jacobian matrix.
IRNDX	I/O: 0, 1, 2, 3, 4	I	1	Residual vector index, incremented by NEQN; NEQN = 1 for RTO_CPSET, RTO_ENSET, RTO_MWSET, RTO_VMSET Input - (1st local residual index)) - 1 Output - Last local residual index
JXNDX	I/O: 0, 1, 2, 3, 4	I	1	Variable vector index, incremented by NVAR; NVAR = 0 for RTO_CPSET, RTO_ENSET, RTO_MWSET, RTO_VMSET Input - No. of variables before addition of RTO_MWSET internal variables. Output - No. of variables after addition of RTO_MWSET internal variables.
NZNDX	I/O: 0, 1, 2, 3, 4	I	1	Jacobian nonzero index, incremented by NNZ; NNZ = NCP + 1 for RTO_MWSET, and NNZ = NCP + 3 for RTO_CPSET, RTO_ENSET, RTO_VMSET Input - (1st local nonzero index) - 1 Output - Last local nonzero index.
ZFLOW	I/O: 3, 4	R	NCP	Component mole fractions. ¹ Input - default composition for total flow below zero tolerance. Output - Composition used for property calculation
ICNAME	O: 1	I	(2,NCP)	Component names.
XNAME	O: 1	R	(3,NVAR)	Variable names.
RNAME	O: 1	R	(3,NEQN)	Function residual name.
IRSTOR	O: 2	I	NNZ	Row index of jacobian.

Variable	ICFLAGS	Type	Dimension	Description
JXSTOR	O: 2	I	NNZ	Column index of jacobian.
RESID	O: 3	R	NEQN	Function residual value.
DRESDX	O: 4	R	NNZ	Jacobian matrix of residual wrt variables.

Notes:

- 1 ZFLOW should be set to the default value of the composition for which the enthalpy should be returned for total flow rates below the minimum flow tolerance. Upon return from the called routine, ZFLOW will contain the composition at which the returned value of the property was calculated. Thus, if the user wishes to use the composition of the previous property call as the default zero flow composition, no adjustment to ZFLOW is necessary between calls.
- 2 The USER3 model main subroutine should include the following common block and statements in order to provide the block name to the physical property calls:

```
#include "rxn_iblkid
      do i=1,2
        iblkid_idblk(i) = ids(i,1)
      enddo
```

Low-Level Physical Property Subroutines

Heat capacity

```
Subroutine rto_cpdim      (ncp, irndx, jxndx, nzndx)
Subroutine rto_cpjac      (ncp, kph, nbopst, idx, flow, pres, temp,
                          cp, lodiag, delta, fscale, pscale,
                          tscale, irndx, jxndx, nzndx, zflow,
                          dresdx)
Subroutine rto_cpnam      (ncp, kph, idx, nlab, irndx, jxndx,
                          nzndx, icname, xname, rname)
Subroutine rto_cpres      (ncp, kph, nbopst, idx, flow, pres,
                          temp, cp, lodiag, irndx, jxndx, nzndx,
                          zflow, resid)
subroutine rto_cpspr      (ncp, jxflow, jxpres, jxtemp, jxcp,
                          irndx, jxndx, nzndx, rstor, jxstor)
```

Enthalpy (mole flow basis):

```
subroutine rto_endim      (ncp, irndx, jxndx, nzndx)
subroutine rto_enjac      (ncp, kph, nbopst, idx, flow, pres,
                          temp, enth, lodiag, delta, fscale,
                          pscale, irndx, jxndx, nzndx, zflow,
                          dresdx)
subroutine rto_ennam      (ncp, kph, idx, nlab, irndx, jxndx,
                          nzndx, icname, xname, rname)
subroutine rto_enres      (ncp, kph, nbopst, idx, flow, pres,
                          temp, enth, lodiag, irndx, jxndx,
                          nzndx, zflow, resid)
```

```

subroutine rto_enspr      (ncp, jxflow, jxpres, jxtemp, jxenth,
                          irndx, jxndx, nzndx, irstor, jxstor)

```

Enthalpy (mole fraction basis):

```

subroutine rto_encdim    (ncp, irndx, jxndx, nzndx)
subroutine rto_encjac    (ncp, kph, nbopst, idx, flow, pres,
                          temp, enth, lodiag, delta, fscale,
                          pscale, irndx, jxndx, nzndx, zflow,
                          dresdx)
subroutine rto_ennam     (ncp, kph, idx, nlab, irndx, jxndx,
                          nzndx, icname, xname, rname)
subroutine rto_encres    (ncp, kph, nbopst, idx, flow, pres,
                          temp, enth, lodiag, irndx, jxndx,
                          nzndx, zflow, resid)
subroutine rto_enspr     (ncp, jxcomp, jxpres, jxtemp, jxenth,
                          irndx, jxndx, nzndx, irstor, jxstor)

```

Molecular Weight:

```

subroutine rto_mwdim     (ncp, irndx, jxndx, nzndx)
subroutine rto_mwjac     (ncp, idx, flow, amw, irndx, jxndx,
                          nzndx, zflow, dresdx)
subroutine rto_mwnam     (ncp, idx, nlab, irndx, jxndx, nzndx,
                          icname, xname, rname)
subroutine rto_mwres     (ncp, idx, flow, amw, irndx, jxndx,
                          nzndx, zflow, resid)
subroutine rto_mwspr     (ncp, jxflow, jxamw, irndx, jxndx,
                          nzndx, irstor, jxstor)

```

Molar Volume:

```

subroutine rto_vmdim     (ncp, irndx, jxndx, nzndx)
subroutine rto_vmjac     (ncp, kph, nbopst, idx, flow, pres,
                          temp, vm, lodiag, delta, fscale,
                          pscale, irndx, jxndx, nzndx, zflow,
                          dresdx)
subroutine rto_vmnam     (ncp, kph, idx, nlab, irndx, jxndx,
                          nzndx, icname, xname, rname)
subroutine rto_vmres     (ncp, kph, nbopst, idx, flow, pres,
                          temp, vm, lodiag, irndx, jxndx, nzndx,
                          zflow, resid)
subroutine rto_vmspr     (ncp, jxflow, jxpres, jxtemp, jxvm,
                          irndx, jxndx, nzndx, irstor, jxstor)

```

Entropy (mole flow basis):

```

subroutine rto_smdim     (ncp, irndx, jxndx, nzndx)
subroutine rto_smjac     (ncp, kph, nbopst, idx, flow, pres,
                          temp, entr, lodiag, delta, fscale,
                          pscale, irndx, jxndx, nzndx, zflow,
                          dresdx)
subroutine rto_smnam     (ncp, kph, idx, nlab, irndx, jxndx,
                          nzndx, icname, xname, rname)

```

```

subroutine rto_smres      (ncp, kph, nbopst, idx, flow, pres,
                          temp, entr, lodiag, irndx, jxndx,
                          nzndx, zflow, resid)
subroutine rto_smspr      (ncp, jxflow, jxpres, jxtemp, jxentr,
                          irndx, jxndx, nzndx, rstor, jxstor)

```

Entropy (mole fraction basis):

```

subroutine rto_smcdim      (ncp, irndx, jxndx, nzndx)
subroutine rto_smcjac      (ncp, kph, nbopst, idx, flow, pres,
                          temp, entr, lodiag, delta, fscale,
                          pscale, irndx, jxndx, nzndx, zflow,
                          dresdx)
subroutine rto_smnam      (ncp, kph, idx, nlab, irndx, jxndx,
                          nzndx, icname, xname, rname)
subroutine rto_smcres      (ncp, kph, nbopst, idx, flow, pres,
                          temp, entr, lodiag, irndx, jxndx,
                          nzndx, zflow, resid)
subroutine rto_smcsprr      (ncp, jxcomp, jxpres, jxtemp, jxentr,
                          irndx, jxndx, nzndx, rstor, jxstor)

```

Other Useful USER3 Utilities

USER3 provides several naming and string manipulation utilities, including:

- U3DNAM Provides default names for unnamed variables and equations.
- U3VNAM Provides a fully qualified name for a block variable.
- U3ENAM Provides a fully qualified name for a block equation.
- U3PKCR Removes embedded blank characters from a string.
- U3CLST Initializes a string to blanks.

USER3 also provides a number of variable attribute manipulation utilities, such as:

- U3FXV Sets status of a variable to fixed.
- U3UFV Sets status of a variable to free.
- U3VFXD Inquire as to a variable's fixed/free status.
- U3VGPT Get the physical type of a variable.
- U3VPPT Put the physical type of a variable.
- U3INCR Set function status to included.
- U3UNCR Set function status to unincluded.
- U3RINC Inquire as to a function's included/unincluded status.

Since the YSTAT and FSTAT arrays have several pieces of information overlayed into each integer location, it is recommended that all status manipulation within the model be done with these utilities rather than by directly working with FSTAT and YSTAT.

Component Object Models (COM)

Aspen Plus has the ability to directly call Component Object Models (COM) to perform unit operation calculations. See Chapter 26, COM Unit Operation Interfaces, beginning on page 269, for detailed information.

6 User Physical Property Models

The Aspen Plus user physical property models allow you to provide models for calculating various major, subordinate, and intermediate physical properties. Aspen Plus allows user models for both conventional and nonconventional properties. You can invoke the user physical property models by selecting the user models for properties on the **Methods | Selected Methods | Models** sheet, or by selecting the user models for nonconventional properties on the **Methods | NC-Props** form. Each user physical property model has a built-in model name, principal subroutine name, and model-specific parameter names.

Note: CALUPP should not be called from user property subroutines. Doing so may result in recursive function calls which may overwrite variables (such as those in Common blocks) used in the first call, which that call may still need. This will lead to unpredictable results.

User Models for Conventional Properties

This section lists the conventional properties calculated by user models. Definitions of the properties, the associated model names, principal subroutine names, and model-specific parameter names are also listed. The following sections explain how to introduce a user conventional property model in a simulation, and how to prepare the necessary principal subroutines for calculating the desired physical properties.

For each physical property, the associated built-in user model name, principal subroutine name, and parameter information are listed in the following table.

Calculated Property	Model Name	Principal Subroutine Name	Parameter Name	Number of Elements	Parameter Type	Definition
\dagger	ESUSR	ESU	ESUA ESUB	5 1	Unary Binary	Equation of state for mixtures
\dagger	ESUSR0	ESU0	ESUA	5	Unary	Equation of state for pure components
\dagger	ESUSR2	ESU2	ESU2A ESU2B	5 1	Unary Binary	Equation of state for mixtures with full analytical derivatives
\dagger	ESUSR20	ESU20	ESU2A	5	Unary	Equation of state for pure components with full analytical derivatives
GAMMA $\dagger\dagger$	GMUSR	GMU	GMUA GMUB	3 2	Unary Binary	Liquid phase activity coefficient
GAMMA $\dagger\dagger$	GMELCUR	GMELCU	GMELUA GMELUB	3 2	Unary Binary	Liquid phase electrolyte system activity coefficient
KVL $\dagger\dagger$	KVLUSR	KVLU	—	—	—	K-value (see K-Value)
HV	HV0USR	HV0U	HV0UA	5	Unary	Pure component vapor molar enthalpy
DHV	DHV0USR	DHV0U	DHV0UA	5	Unary	Pure component vapor molar enthalpy departure
HVMX	HV2USR	HV2U	HV2UA	5	Unary	Mixture vapor molar enthalpy
HVMX	HV2USR2	HV2U2	HV2U2A	5	Unary	Mixture vapor molar enthalpy $\dagger\dagger\dagger$
DHVMX	DHV2USR	DHV2U	DHV2UA	5	Unary	Mixture vapor molar enthalpy departure
HL	HL0USR	HL0U	HL0UA	5	Unary	Pure component liquid molar enthalpy
DHL	DHL0USR	DHL0U	DHL0UA	5	Unary	Pure component liquid molar enthalpy departure
HLMX	HL2USR	HL2U	HL2UA	5	Unary	Mixture liquid molar enthalpy
HLMX	HL2USR2	HL2U2	HL2U2A	5	Unary	Mixture liquid molar enthalpy $\dagger\dagger\dagger$

Calculated Property	Model Name	Principal Subroutine Name	Parameter Name	Number of Elements	Parameter Type	Definition
DHLMX	DHL2USR	DHL2U	DHL2UA	5	Unary	Mixture liquid molar enthalpy departure
HS	HS0USR	HS0U	HS0UA	5	Unary	Pure component solid molar enthalpy
DHS	DHS0USR	DHS0U	DHS0UA	5	Unary	Pure component solid molar enthalpy departure
HSMX	HS2USR	HS2U	HS2UA	5	Unary	Mixture solid molar enthalpy
HSMX	HS2USR2	HS2U2	HS2U2A	5	Unary	Mixture solid molar enthalpy ⁺⁺⁺
DHSMX	DHS2USR	DHS2U	DHS2UA	5	Unary	Mixture solid molar enthalpy departure
DHVL	DHVLUSR	DHVLU	DHVLUA	5	Unary	Molar enthalpy of vaporization
DHLS	DHLSUSR	DHLSU	DHLSUA	5	Unary	Molar enthalpy of fusion
DHVS	DHVSUSR	DHVSU	DHVSUA	5	Unary	Molar enthalpy of sublimation
PHIL ⁺⁺	PHL0USR	PHL0U	PHL0UA	5	Unary	Pure component liquid fugacity coefficient
PHILMX ⁺⁺	PHL1USR	PHL1U	PHL1UA	5	Unary	Fugacity coefficient of a component in a liquid mixture
PHIV ⁺⁺	PHV0USR	PHV0U	PHV0UA	5	Unary	Pure component vapor fugacity coefficient
PHIVMX ⁺⁺	PHV1USR	PHV1U	PHV1UA	5	Unary	Fugacity coefficient of a component in a vapor mixture
PHIS ⁺⁺	PHS0USR	PHS0U	PHS0UA	5	Unary	Pure component solid fugacity coefficient
PHISMX ⁺⁺	PHS1USR	PHS1U	PHS1UA	5	Unary	Fugacity coefficient of a component in a solid mixture
PL ⁺⁺	PL0USR	PL0U	PL0UA	10	Unary	Liquid vapor pressure
PS ⁺⁺	PS0USR	PS0U	PS0UA	5	Unary	Solid vapor pressure
VV	VV0USR	VV0U	VV0UA	5	Unary	Pure component vapor molar volume
VVMX	VV2USR	VV2U	VV2UA	5	Unary	Mixture vapor molar volume
VVMX	VV2USR2	VV2U2	VV2U2A	5	Unary	Mixture vapor molar volume ⁺⁺⁺
VL	VL0USR	VL0U	VL0UA	5	Unary	Pure component liquid molar volume
VLPM	VL1USR	VL1U	VL1UA	5	Unary	Partial molar liquid volume
VLMX	VL2USR	VL2U	VL2UA	5	Unary	Mixture liquid molar volume
VLMX	VL2USR2	VL2U2	VL2U2A	5	Unary	Mixture liquid molar volume ⁺⁺⁺
VS	VS0USR	VS0U	VS0UA	5	Unary	Pure component solid molar volume

Calculated Property	Model Name	Principal Subroutine Name	Parameter Name	Number of Elements	Parameter Type	Definition
VSMX	VS2USR	VS2U	VS2UA	5	Unary	Mixture solid molar volume
VSMX	VS2USR2	VS2U2	VS2U2A	5	Unary	Mixture solid molar volume ⁺⁺⁺
MUV	MUV0USR	MUV0U	MUV0UA	5	Unary	Pure component vapor viscosity
MUVMX	MUV2USR	MUV2U	MUV2UA	5	Unary	Mixture vapor viscosity
MUVMX	MUV2USR2	MUV2U2	MUV2U2A	5	Unary	Mixture vapor viscosity ⁺⁺⁺
MUL	MUL0USR	MUL0U	MUL0UA	5	Unary	Pure component liquid viscosity
MULMX	MUL2USR	MUL2U	MUL2UA	5	Unary	Mixture liquid viscosity
MULMX	MUL2USR2	MUL2U2	MUL2U2A	5	Unary	Mixture liquid viscosity ⁺⁺⁺
KV	KV0USR	KV0U	KV0UA	5	Unary	Pure component vapor thermal conductivity
KVMX	KV2USR	KV2U	KV2UA	5	Unary	Mixture vapor thermal conductivity
KVMX	KV2USR2	KV2U2	KV2U2A	5	Unary	Mixture vapor thermal conductivity ⁺⁺⁺
KL	KL0USR	KL0U	KL0UA	5	Unary	Pure component liquid thermal conductivity
KLMX	KL2USR	KL2U	KL2UA	5	Unary	Mixture liquid thermal conductivity
KLMX	KL2USR2	KL2U2	KL2U2A	5	Unary	Mixture liquid thermal conductivity ⁺⁺⁺
KS	KS0USR	KS0U	KS0UA	5	Unary	Pure component solid thermal conductivity
KSMX	KS2USR	KS2U	KS2UA	5	Unary	Mixture solid thermal conductivity
KSMX	KS2USR2	KS2U2	KS2U2A	5	Unary	Mixture solid thermal conductivity ⁺⁺⁺
DV	DV0USR	DV0U	DV0UA	5	Unary	Binary diffusion coefficients of vapor
DL	DL0USR	DL0U	DL0UA	5	Unary	Binary diffusion coefficients of liquid
DVMX	DV1USR	DV1U	DV1UA	5	Unary	Diffusion coefficient of a component in a vapor mixture
DLMX	DL1USR	DL1U	DL1UA	5	Unary	Diffusion coefficient of a component in a liquid mixture
SIGL	SIG0USR	SIG0U	SIG0UA	5	Unary	Pure component surface tension
SIGLMX	SIG2USR	SIG2U	SIG2UA	5	Unary	Mixture surface tension
SIGLMX	SIG2USR2	SIG2U2	SIG2U2A	5	Unary	Mixture surface tension ⁺⁺⁺
GLMX	GL2USR	GL2U	GL2UA	5	Unary	Mixture liquid molar Gibbs free energy

Calculated Property	Model Name	Principal Subroutine Name	Parameter Name	Number of Elements	Parameter Type	Definition
GLMX	GL2USR2	GL2U2	GL2U2A	5	Unary	Mixture liquid molar Gibbs free energy ^{†††}
GVMX	GV2USR	GV2U	GV2UA	5	Unary	Mixture vapor molar Gibbs free energy
GVMX	GV2USR2	GV2U2	GV2U2A	5	Unary	Mixture vapor molar Gibbs free energy ^{†††}
GSMX	GS2USR	GS2U	GS2UA	5	Unary	Mixture solid molar Gibbs free energy
GSMX	GS2USR2	GS2U2	GS2U2A	5	Unary	Mixture solid molar Gibbs free energy ^{†††}
DSSMX	DSS2USR	DSS2U	DSS2UA	5	Unary	Mixture solid molar entropy departure
DSLTX	DSL2USR	DSL2U	DSL2UA	5	Unary	Mixture liquid molar entropy departure
DSVMX	DSV2USR	DSV2U	DSV2UA	5	Unary	Mixture vapor molar entropy departure
DGSMX	DGS2USR	DGS2U	DGS2UA	5	Unary	Mixture solid molar Gibbs free energy departure
DGLMX	DGL2USR	DGL2U	DGL2UA	5	Unary	Mixture liquid molar Gibbs free energy departure
DGVMX	DGV2USR	DGV2U	DGV2UA	5	Unary	Mixture vapor molar Gibbs free energy departure
DGS	DGS0USR	DGS0U	DGS0UA	5	Unary	Pure component solid molar Gibbs free energy departure
GS	GS0USR	GS0U	GS0UA	5	Unary	Pure component solid molar Gibbs free energy
DGL	DGL0USR	DGL0U	DGL0UA	5	Unary	Pure component liquid molar Gibbs free energy departure
GL	GL0USR	GL0U	GL0UA	5	Unary	Pure component liquid molar Gibbs free energy
DGV	DGV0USR	DGV0U	DGV0UA	5	Unary	Pure component vapor molar Gibbs free energy departure
GV	GV0USR	GV0U	GV0UA	5	Unary	Pure component vapor molar Gibbs free energy

[†]ESUSR and ESUSR0 are the model names for user equation-of-state models. All of the built-in equation-of-state models are capable of calculating and returning fugacity coefficients, enthalpy, entropy, and Gibbs free energy departures, and molar volumes for mixtures or pure components in the vapor or liquid phases. While this is not required of a user equation-of-state model, you should include those properties that will actually be required when the model is called by a physical property monitor.

^{††}The quantity calculated and returned should be the natural logarithm of the property.

^{†††}These mixture property models are intended for users to develop special mixing rules to calculate mixture properties from the constituent pure component properties calculated by a specified route.

Principal User Model

Subroutines for Conventional Properties

For each user model, you must provide a principal subroutine that calculates and returns the desired physical properties. Since the principal subroutines are called directly by the appropriate physical property monitors, they have a fixed name and argument list structure. The principal subroutine can call any additional subroutine. There is no restriction on the argument lists and the number of these additional subroutines.

Calling Sequence for Pure Component Properties

```
SUBROUTINE subrname      (T, PI, N, IDX, IRW, IIW, KCALC, KOP,  
                          NDS, KDIAG, QI, DQI, KER)
```

Principal subroutine names: HV0U, DHV0U, HL0U, DHL0U, HS0U, DHS0U, PHL0U, PHV0U, PHS0U, PL0U, PS0U, VV0U, VL0U, VS0U, MUV0U, MUL0U, KV0U, KL0U, KS0U, SIG0U, DHLSU, DHVSU, DHVLU, DGS0U, GS0U, DGL0U, GL0U, DGV0U, GV0U

Calling Sequence for Properties of Components in a Mixture

```
SUBROUTINE subrname      (T, P, Z, N, IDX, IRW, IIW, KCALC,  
                          KOP, NDS, KDIAG, QIMX, DQIMX, KER)
```

Principal subroutine names: PHL1U, PHV1U, PHS1U, VL1U, DV1U, DL1U

Calling Sequence for User Activity Coefficient Model

```
SUBROUTINE GMU           (T, P, Z, N, IDX, IRW, IIW, KCALC,  
                          KOP, NDS, KDIAG, QIMX, DQIMX, KER,  
                          NSUB, NSUP, IDXSUB, IDXSUP, GAMSC,  
                          DGAMSC)
```

Calling Sequence for User Electrolyte Activity Coefficient Model

```
SUBROUTINE GMUELC        (T, P, X, N, NSUB, NSUP, NCAT, NANI,  
                          IDX, IDXN, IDXSUB, IDXSUP, IDXCAT,  
                          IDXANI, IRW, IIW, KCALC, NDS, KDIAG,  
                          GAMA, GAMSC, DGAMA, DGAMSC, KER, KOP)
```

(See Electrolyte Calculations)

Calling Sequence for Properties of a Mixture

```
SUBROUTINE subrname      (T, P, Z, N, IDX, IRW, IIW, KCALC,  
                          KOP, NDS, KDIAG, QMX, DQMX, KER)
```

Principal subroutine names: HV2U, DHV2U, HL2U, DHL2U, HS2U, DHS2U, VV2U, VL2U, VS2U, MUV2U, MUL2U, KV2U, KL2U, KS2U, SIG2U, GL2U, GV2U, GS2U, DSS2U, DSL2U, DSV2U, DGS2U, DGL2U, DGV2U

Calling Sequence for Properties of a Mixture That Depends on Pure Component Properties and User Mixing Rule

```
SUBROUTINE subrname      (T, P, Z, N, IDX, XMW, SG, VLSTD,
                          PARAM, QI, DQI, DPQI, KSW, KOP, NDS,
                          KDIAG, QMX, DQMX, DPQMX, KER)
```

Principal subroutine names: HV2U2, HL2U2, HS2U2, VV2U2, VL2U2, VS2U2, MUV2U2, MUL2U2, KV2U2, KL2U2, KS2U2, SIG2U2, GL2U2, GV2U2, GS2U2

Thermodynamic and transport properties of mixtures are generally a non-linear function of the constituent pure component properties. The above models and subroutines allow users to provide their own mixing rules to calculate these properties from pure component properties that are computed using a specified route.

Calling Sequence for User EOS Subroutine (ESU)

```
SUBROUTINE ESU           (T, P, Z, N, IDX, IRW, IIW, KVL, KPHI,
                          KH, KS, KG, KV, NDS, KDIAG, PHIMX,
                          DHMX, DSMX, DGMX, VMX, DPHIMX, DDHMX,
                          DDSMX, DDGMX, DVMX, KER, KOP)
```

Calling Sequence for User Pure Component EOS Subroutine (ESU0)

```
SUBROUTINE ESU0         (T, PI, N, IDX, IRW, IIW, KVL, KPHI,
                          KH, KS, KG, KV, NDS, KDIAG, PHI, DH,
                          DS, DG, V, DPHI, DDH, DDS, DDG, DV,
                          KER, KOP)
```

Calling Sequence for User EOS Subroutine with Full Analytical Derivatives (ESU2)

```
SUBROUTINE ESU2         (T, P, Z, N, IDX, RW, IW, KVL, NDS,
                          KDIAG, KPHI, KH, KS, KG, KV, PHIMX,
                          DPHIMXDT, DPHIMXDP, DPHIMXDN, DPHIMXDX,
                          DHMX, DDHMXDT, DDHMXDP, DDHMXDN, DDHMXDX,
                          DSMX, DDSMXDT, DDSMXDP, DDSMXDN, DDSMXDX,
                          DGMX, DDSMXDT, DDGMXDP, DDGMXDN, DDGMXDX,
                          VMX, DVMXDT, DVMXDP, DVMXDN, DVMXDX,
                          KER, KOP)
```

Calling Sequence for User Pure Component EOS Subroutine with Full Analytical Derivatives (ESU20)

```
SUBROUTINE ESU20        (T, P, N, IDX, RW, IW, KVL, NDS,
                          KDIAG, KER, KPHI, KH, KS, KG, KV, PHI,
                          DPHIDT, DPHIDP, DH, DDHDT, DDHDP, DS,
```

DDSDT, DDSDP, DG, DDGDT, DDGDP, V, DVDT,
DVDP, KOP)

Calling Sequence for User K-Value Subroutine (KVLU)

SUBROUTINE KVLU (T, P, X, Y, N, IDX, IRW, IIW, KCALC,
KOP, NDS, KDIAG, XK, DXK, KER)

Argument Descriptions: Conventional Property Subroutines

Variable	I/O [†]	Type	Dimension	Description
T	I	REAL*8	—	Temperature (K)
P	I	REAL*8	—	Pressure (N/m ²)
PI	I	REAL*8	N	Vector of pressures for pure component properties (either system pressure or vapor pressure) at which the calculations should be performed (N/m ²)
Z	I	REAL*8	N	Component mole fraction vector (see X, Y, Z)
X	I	REAL*8	N	Liquid mole fraction vector (see X, Y, Z)
Y	I	REAL*8	N	Vapor mole fraction vector (see X, Y, Z)
N	I	INTEGER	—	Number of components present
IDX	I	INTEGER	N	Component index vector (see IDX)
NSUB	I	INTEGER	—	Number of subcritical components within IDX
NSUP	I	INTEGER	—	Number of supercritical components within IDX
NCAT	I	INTEGER	—	Number of cations within IDX
NANI	I	INTEGER	—	Number of anions within IDX
IDXSUB	I	INTEGER	NSUB	Subcritical component index vector (see Partial Component Index Vectors)
IDXSUP	I	INTEGER	NSUP	Supercritical component index vector (see Partial Component Index Vectors)
IDXM	I	INTEGER	NSUB+NSUP	Molecular component index vector (see Partial Component Index Vectors)
IDXCAT	I	INTEGER	NCAT	Cation component index vector (see Partial Component Index Vectors)
IDXANI	I	INTEGER	NANI	Anion component index vector (see Partial Component Index Vectors)
IRW	I	INTEGER	—	Offset for real work array (see Real and Integer Work Areas)
IIW	I	INTEGER	—	Offset for integer work array (see Real and Integer Work Areas)
RW	I/O	REAL*8	see note	Real Work Area (see Real and Integer Work Areas)
IW	I/O	INTEGER	see note	Integer Work Area (see Real and Integer Work Areas)

Variable	I/O [†]	Type	Dimension	Description
KCALC	I	INTEGER	—	Calculation code: KCALC=1, Calculate property only KCALC=2, Calculate temperature derivative of property only KCALC=3, Calculate property and temperature derivative.
KSW	I	INTEGER	3	Calculation code for user mixing rules. For each element, 0 = Do not calculate, 1 = Calculate: KSW(1): The property itself KSW(2): Temperature derivative of property KSW(3): Pressure derivative of property
XMW	I	REAL*8	NCC **	Molecular weight vector
SG	I	REAL*8	NCC **	Specific gravity vector
VLSTD	I	REAL*8	NCC **	Standard liquid volume vector (m ³ /kgmole)
PARAM	I	REAL*8	5*NCC **	Vector of user mixing rule model parameter, such as HL2U2A parameter for HL2U2 model.
KOP	I	INTEGER	10	Integer vector containing up to 10 model option codes (see KOP)
NDS	I	INTEGER	—	Data set number of the model-specific parameters (see Parameter Retrieval, this chapter, for explanation of how to use this number to retrieve the correct set of parameters.)
KDIAG	I	INTEGER	—	Diagnostic level code for the control of warning and error messages. (see KDIAG)
KVL	I	INTEGER	—	Vapor - liquid flag: KVL=1, vapor. KVL=2, liquid
KPHI	I	INTEGER	see note	Calculation code for fugacity coefficients (see Calculation Codes)
KH	I	INTEGER	see note	Calculation code for the enthalpy departure (see Calculation Codes)
KS	I	INTEGER	see note	Calculation code for the entropy departure (see Calculation Codes)
KG	I	INTEGER	see note	Calculation code for the Gibbs energy departure (see Calculation Codes)
KV	I	INTEGER	see note	Calculation code for the molar volume (see Calculation Codes)
QI	I/O	REAL*8	N	Vector of pure component properties. For user mixing rules, this is an input variable; for all other models it is an output variable.
DQI	I/O	REAL*8	N	Vector of temperature derivatives of pure component properties. For user mixing rules, this is an input variable; for all other models it is an output variable.
DPQI	I	REAL*8	N	Vector of pressure derivatives of pure component properties for user mixing rules

Variable	I/O ⁺	Type	Dimension	Description
QIMX	O	REAL*8	N	Vector of properties of components in a mixture
DQIMX	O	REAL*8	N	Vector of temperature derivatives of properties of components in a mixture
QMX	O	REAL*8	—	Mixture property
DQMX	O	REAL*8	—	Temperature derivative of a mixture property
DPQMX	O	REAL*8	—	Pressure derivative of a mixture property
GAMSC	O	REAL*8	NSUB, NSUP	Natural logarithm of the activity coefficients of supercritical components at infinite dilution in subcritical components
DGAMSC	O	REAL*8	NSUB, NSUP	Temperature derivative of GAMSC
GAMA	O	REAL*8	N	Natural logarithm of the activity coefficients of all components; for ionic species, this is the natural logarithm of the unsymmetric activity coefficient (see Electrolyte Calculations)
DGAMA	O	REAL*8	N	Temperature derivative of GAMA (see Electrolyte Calculations)
PHI	O	REAL*8	N	Vector of logarithms for fugacity coefficients of pure components
DPHI	O	REAL*8	N	Vector of temperature derivatives of PHI
DPHIDT	O	REAL*8	N	Vector of temperature derivatives of PHI
DPHIDP	O	REAL*8	N	Vector of pressure derivatives of PHI
PHIMX	O	REAL*8	N	Vector of logarithms for fugacity coefficients of components in a mixture
DPHIMX	O	REAL*8	N	Vector of temperature derivative of PHIMX
DPHIMXDT	O	REAL*8	N	Vector of temperature derivative of PHIMX
DPHIMXDP	O	REAL*8	N	Vector of pressure derivative of PHIMX
DPHIMXDN	O	REAL*8	N,N	Vector of molar flow derivative of PHIMX $DPHIMXDN(i,j) = dPHIMX(i)/dN(j)$
DPHIMXDX	O	REAL*8	N,N	Vector of composition derivative of PHIMX $DPHIMXDX(i,j) = dPHIMX(i)/dX(j)$
DH	O	REAL*8	N	Vector of pure component enthalpy departures (J/kgmole)
DDH	O	REAL*8	N	Vector of temperature derivatives of DH (J/kgmole-K)
DDHDT	O	REAL*8	N	Vector of temperature derivatives of DH
DDHDP	O	REAL*8	N	Vector of pressure derivatives of DH
DHMX	O	REAL*8	—	Enthalpy departure of a mixture (J/kgmole)
DDHMX	O	REAL*8	—	Temperature derivative of DHMX (J/kgmole-K)
DDHMXDT	O	REAL*8	—	Temperature derivative of DHMX
DDHMXDP	O	REAL*8	—	Pressure derivative of DHMX
DDHMXDN	O	REAL*8	N	Vector of molar flow derivative of DHMX
DDHMXDX	O	REAL*8	N	Vector of composition derivative of DHMX

Variable	I/O [†]	Type	Dimension	Description
DS	O	REAL*8	N	Vector of pure component entropy departures (J/kgmole-K)
DDS	O	REAL*8		Vector of temperature derivatives of DS (J/kgmole-K)
DDSDT	O	REAL*8	N	Vector of temperature derivatives of DS
DDSDP	O	REAL*8	N	Vector of pressure derivatives of DS
DSMX	O	REAL*8	—	Entropy departure of a mixture (J-kgmole-K)
DDSMX	O	REAL*8	—	Temperature derivative of DSMX (J-kgmole-K ²)
DDSMXDT	O	REAL*8	—	Temperature derivative of DSMX
DDSMXDP	O	REAL*8	—	Pressure derivative of DSMX
DDSMXDN	O	REAL*8	N	Vector of molar flow derivative of DSMX
DDSMXDX	O	REAL*8	N	Vector of composition derivative of DSMX
DG	O	REAL*8	N	Vector of pure component free energy departures (J/kgmole)
DDG	O	REAL*8	N	Vector for temperature derivatives of DG (J/kgmole-K)
DDGDT	O	REAL*8	N	Vector of temperature derivatives of DG
DDGDP	O	REAL*8	N	Vector of pressure derivatives of DG
DGMX	O	REAL*8	—	Gibbs energy departure of a mixture (J/kgmole)
DDGMX	O	REAL*8	—	Temperature derivative of DGMX (J/kgmole)
DDGMXDT	O	REAL*8	—	Temperature derivative of DGMX
DDGMXDP	O	REAL*8	—	Pressure derivative of DGMX
DDGMXDN	O	REAL*8	N	Vector of molar flow derivative of DGMX
DDGMXDX	O	REAL*8	N	Vector of composition derivative of DGMX
V	O	REAL*8	N	Vector of pure component molar volumes (m ³ /kgmole)
DV	O	REAL*8	N	Vector of temperature derivatives of V (m ³ /kgmole-K)
DVDT	O	REAL*8	N	Vector of temperature derivatives of V
DVDP	O	REAL*8	N	Vector of pressure derivatives of V
VMX	O	REAL*8	—	Molar volume of a mixture (m ³ /kgmole)
DVMX	O	REAL*8	—	Temperature derivative of VMX (m ³ /kgmole-K)
DVMXDT	O	REAL*8	—	Temperature derivative of VMX
DVMXDP	O	REAL*8	—	Pressure derivative of VMX
DVMXDN	O	REAL*8	N	Vector of molar flow derivative of VMX
DVMXDX	O	REAL*8	N	Vector of composition derivative of VMX
XK	O	REAL*8	N	Vector of K-values
DXK	O	REAL*8	N	Vector of temperature derivative of K-values
KER	O	INTEGER	—	Error return code (not used)

[†] I = Input to subroutine, O = Output from subroutine

^{††} NCC is the number of conventional components in a simulation. See Appendix A, COMMON DMS_NCOMP.

IDX

IDX is a vector containing component numbers of the conventional components actually present in the order that they appear on the **Components | Specifications | Selection** sheet. For example, if only the first and third conventional components are present, then N=2 and IDX=(1,3). For component properties, output vectors should be filled in for only the components actually present.

Partial Component Index Vectors

IDXSUB and IDXSUP are vectors containing the locations within IDX that point to subcritical or supercritical components, respectively. For example, if the first component in IDX is subcritical, and the second is supercritical:

```
IDXSUB(1) = 1  
IDSUP(1) = 2
```

To get the index K of the component in the order it appears on the **Components | Specification | Selection** sheet:

```
KSUB = IDX(IDXSUB(1))  
KSUP = IDX(IDXSUP(1))
```

You declare supercritical components on the **Components | Henry Comps** form.

IDXM is a vector of all molecular (non-ionic) components and includes all of the components in IDXSUB and IDSUP. IDXCAT and IDXANI are vectors of the cation and anion components, respectively. These vectors are used in the same manner as IDXSUB and IDSUP.

X, Y, Z

X, Y, and Z are packed mole fraction vectors of length N containing the mole fractions of the components that are present and are to be included in the calculations. The order of components is defined by the corresponding IDX vector. For example, if N=2 and IDX=(1,3), Y(2) is the vapor mole fraction of the conventional component listed third on the **Components | Specifications | Selection** sheet.

Real and Integer Work Areas

To access the real and integer work areas reserved for a user physical property model using IRW and IIW:

- 1 Include labeled commons DMS_PPWORK and DMS_IPWORK. Include directives begin in column 1.

```
#include "dms_ppwork.cmn"  
#include "dms_ipwork.cmn"
```

- 2 Include labeled common DMS_PLEX. Include directives begin in column 1.

```
#include "dms_plex.cmn"
```

- 3 Add the following declarations:

```
REAL*8 B(1)
EQUIVALENCE (B(1), IB(1))
```

The first element of the real work area that the user model can use is:

```
B(PPWORK_LPPWK + IRW + 1)
```

The length of this area is:

```
10 + 4 * NCOMP_NCC + NCOMP_NCC * NCOMP_NCC
```

where NCOMP_NCC is the total number of conventional components declared in the simulation (see Appendix A, COMMON DMS_NCOMP).

The first element of the integer work area that the user model can use is:

```
IB(IPWORK_LIPWK + IIW + 1)
```

The length of this area is 20 elements. Both the integer and real work areas are local to the user subroutine, and their contents are not saved/retained from one call to the next.

Some subroutines use work area arguments RW and IW instead, directly providing the variables for these work areas.

KOP

You can use the 10-element option code vector, KOP, to provide additional calculational flexibility within your user model. For example, multiple correlations within the same user model may be handled by branching to different segments of a subroutine, or to different subroutines, depending on the option codes. Option codes default to zero for all user models. To assign specific integer values to the option codes, go to the **Methods | Specified Methods | Models** sheet of the property method of interest. Select the user property model that you want to modify the option codes, and click the Option Codes button.

KDIAG

You can use KDIAG to control the printing of error, warning, and diagnostic information in the History File. The values for KDIAG correspond to the values for message levels described in the Help screens for Physical Properties on the **Setup | Specifications | Diagnostics** sheet or the **Block Options | Diagnostics** sheet of a unit operation model. If KDIAG=N, all messages at level N and below will be printed.

Calculation Codes

The calculation code KCALC (and KPHI, KH, KS, KG, KV in ESU and ESU0) can have four values:

- 0 = Do not calculate the property or its temperature derivative
- 1 = Calculate the property only
- 2 = Calculate the temperature derivative of the property only
- 3 = Calculate the property and its temperature derivative

The calculation code KSW of user mixing rules has three elements. KPHI, KH, KS, KG, KV have three elements in ESU20 and six elements in ESU2 with the 4th currently unused. The elements are used to indicate whether to calculate these properties:

Element 1: The property itself
Element 2: The temperature derivative of property
Element 3: The pressure derivative of property
Element 5: The molar flow derivative of property
Element 6: The composition derivative of property

For each element, the value 0 means do not calculate; value 1 (or otherwise greater than zero) means calculate.

Range of Applicability

A user model should test for the range of applicability of the model, and provide a reasonable extrapolation beyond the range. A warning message should be written to the History File. Frequently, the reasonable extrapolation can be a linear extrapolation with a slope equal to that of the correlation at or near the appropriate bound. When the correlation bounds are violated or when any other condition occurs that may make the results questionable, an error or warning message should be written to the History File. (See Chapter 4, Aspen Plus Error Handler.)

Units of Measurement

All quantities passed to a principal subroutine through its argument list and through the universal constant labeled commons are in SI units. Model-specific parameters may be in any units. All calculated properties returned through the argument list must be in SI.

Global Physical Property Constants

The universal gas constant and the ideal gas reference pressure and temperature are available in SI units in COMMON /PPUTL_PPGLOB/ (see Appendix A).

User K-Value

Aspen Plus treats the user K-value as if it were a liquid fugacity coefficient model. To invoke the user K-value model for K-value calculations, you must modify your current property method using the **Methods | Specified Methods | Routes** sheet. On this sheet, select

- KVLUSR as the Route ID for the property PHILMX.
- PHIVMX00 as the Route ID for the property PHIVMX.

Electrolyte Calculations

Electrolyte user models should always calculate properties on a true-component basis for all species. If apparent component calculations are requested, the Aspen Physical Property System will make the necessary conversions of the data returned by the user model.

Model-Specific Parameters for Conventional Properties

In general, a user model may require as input model-specific parameters and universal constants such as critical temperature. The universal constants presently defined in Aspen Plus are listed in this section. You can access any of these universal constants by

- Including the labeled common DMS_PLEX in the model.
- Determining the offset of the parameter into the PLEX by calling the utility DMS_IFCMN (or DMS_IFCMNC). (See Chapter 4.)

Each user model has a built-in model-specific unary parameter and, in some cases, a built-in model-specific binary parameter as well (see the User Models for Conventional Properties table, this chapter). These parameters are stored in the labeled common DMS_PLEX, and can be accessed by first determining their offset in the PLEX.

In user mixing rules there are 4 unary parameters as arguments in the principal user model subroutines:

- XMW (molecular weight)
- SG (specific gravity)
- VLSTD (standard liquid volume)
- PARAM (built-in model-specific parameter)

If additional model-specific parameters are required, you can introduce them by assigning names to them on the **Customize | User Parameters** form. These parameters will be stored in the labeled common DMS_PLEX, and can be accessed by first determining their offset in the PLEX.

If parameters required by a user model are already known to Aspen Plus because of their association with built-in Aspen Plus models, but are not required by any models invoked by the user, you must include these parameters on the **Customize | User Parameters** form.

All binary user model parameters default to zero, and all unary parameters default to RGLOB_RMISS (from COMMON/DMS_RGLOB/). You can enter values of the universal constants and model-specific parameters on the **Methods | Parameters** forms.

Universal Constant Names and Definitions

Name	Definition
MW	Molecular weight (kg/kgmole)

Name	Definition
TC	Critical temperature (K)
PC	Critical pressure (N/m ²)
VC	Critical volume (m ³ /kgmole)
OMEGA	Pitzer acentric factor
DHFORM	Standard heat of formation of ideal gas at 298.15 K (J/kgmole)
DGFORM	Standard free energy of formation of ideal gas at 298.15 K (J/kgmole)
TB	Normal boiling point (K)
VB	Liquid molar volume at TB (m ³ /kgmole)
TFP	Normal freezing point (K)
DELTA	Solubility parameter at 298.15 K (J/ m ³) ^{1/2}
MUP	Dipole moment (J * m ³) ^{1/2}
RGYR	Radius of gyration (m)
CHI	Stiel polar factor
CHARGE	Ionic charge number (positive for cations, negative for anions, zero for molecular species)

Naming Model-Specific Parameters

Choose the name of the additional model-specific parameters to avoid conflict with the built-in Aspen Plus names. We recommend using the names that are listed in the User Models for Conventional Properties table, this chapter, with the last character changed to B, C, and so on.

Multiple Data Sets

You can define multiple data sets for the model-specific parameters—both those built-in and those defined on the **Customize | User Parameters** form.

For built-in parameters that allow multiple data sets, use the **Methods | Selected Methods | Models** sheet to specify the data set number (greater than 1) for the model and its parameters.

For user-defined parameters, you must:

- Use the **Customize | User Parameters** form to define the parameter and the maximum number of data sets allowed.
- Use the **Methods | Selected Methods | Models** sheet to specify the user model and the data set number (greater than 1) for the model and its parameters.

Example: Accessing Critical Temperature

The user model requires critical temperature. In the user subroutine, include:

```
#include "dms_plex.cmn"
      INTEGER DMS_IFCMNC
      INTEGER LTC
      .
      .
```



```

      .
      LTC=DMS_IFCMNC( 'TC' )

```

The TC values start at B(LTC + 1)

Example: Accessing a 2-element Binary Parameter

For the liquid phase activity coefficient model, GMUSR, use both the 3-element unary parameter GMUA and the 2-element binary parameter GMUB. In the user subroutine, include:

```

#include "dms_plex.cmn"
      INTEGER DMS_IFCMNC
      INTEGER LGMUA, LGMUB
      REAL*8 B(1)
      EQUIVALENCE (B(1), IB(1))
      .
      .
      .
      LGMUA=DMS_IFCMNC( 'GMUA' )
      LGMUB=DMS_IFCMNC( 'GMUB' )

```

The parameter values start at B(LGMUA + 1) and B(LGMUB + 1)

Example: Accessing a 3-element Binary Parameter

Same as example for 2-element binary parameter; however, the user model requires an additional 3-element binary parameter with a maximum of 2 data sets. Specify on the **Customize | User Parameters** form:

Parameter Name: GMUC

Parameter Type: Binary Parameters

No. of Elements: 3

Maximum Data Sets: 2

In the user subroutine, include:

```

#include "dms_plex.cmn"
      INTEGER DMS_IFCMNC
      INTEGER LGMUC
      REAL*8 B(1)
      EQUIVALENCE (B(1), IB(1))
      .
      .
      .
      LGMUC=DMS_IFCMNC( 'GMUC' )

```

The parameter values start at B(LGMUC + 1)

Parameter Retrieval

Since parameter areas are not packed, Aspen Plus can retrieve parameter values for the subset of N components included in the calculation using the index vector IDX. To determine the PLEX location K of the IELth element of

the NDStH data set of the unary parameter GMUA (which has 3 elements) for the Ith component in the packed IDX array:

```

INTEGER DMS_IFCMNC
...
NEL = 3
LGMUA = DMS_IFCMNC('GMUA')
LNDS = LGMUA+NEL*NCOMP_NCC*(NDS-1)
LI= LNDS+NEL*(IDX(I)-1)
K = LI+IEL

```

Variable name	Description
NEL	Number of elements for GMUA
LGMUA	PLEX Offset of GMUA
LNDS	PLEX offset of the NDStH data set for GMUA
LI	PLEX offset of the Ith component within the NDStH data set for GMUA
K	PLEX location of the IELth element of GMUA for the Ith component within the NDStH data set for GMUA
NCOMP_NCC	Number of conventional components in the simulation, stored in labeled common /DMS_NCOMP/ (see Appendix A)

Variable name	Description
NEL	Number of elements for GMUA
LGMUA	PLEX Offset of GMUA
LNDS	PLEX offset of the NDStH data set for GMUA
LI	PLEX offset of the Ith component within the NDStH data set for GMUA
K	PLEX location of the IELth element of GMUA for the Ith component within the NDStH data set for GMUA
NCOMP_NCC	Number of conventional components in the simulation, stored in labeled common /DMS_NCOMP/ (see Appendix A)

To determine the PLEX location K of the IELth element of the NDStH data set of the binary parameter GMUB for the component pair (i, j):

```

INTEGER DMS_IFCMNC
...
NEL = 2
LGMUB = DMS_IFCMNC('GMUB')
LNDS = LGMUB+NEL*NCOMP_NCC*NCOMP_NCC*(NDS-1)
LIJ = LNDS+NEL*NCOMP_NCC*(IDX(J)-1)+NEL*(IDX(I)-1)
K = LIJ+IEL

```

Variable name	Description
NEL	Number of elements for GMUB
LGMUB	PLEX offset of GMUB
LNDS	PLEX offset of the NDStH data set for GMUB
LIJ	PLEX offset of GMUB for the (i, j) binary
K	PLEX location of the IELth element of GMUB for the (i, j) binary
NCOMP_NCC	Number of conventional components in the simulation, stored in labeled common /DMS_NCOMP/ (see Appendix A)

Variable name	Description
NEL	Number of elements for GMUB
LGMUB	PLEX offset of GMUB
LNDS	PLEX offset of the NDStH data set for GMUB
LIJ	PLEX offset of GMUB for the (i, j) binary
K	PLEX location of the IELth element of GMUB for the (i, j) binary
NCOMP_NCC	Number of conventional components in the simulation, stored in labeled common /DMS_NCOMP/ (see Appendix A)

User Models for Nonconventional Properties

Aspen Plus provides two user models for calculating nonconventional properties, one for enthalpy and the other for density. Their built-in model names, properties, subroutine names, and parameter names are listed below.

You can use the **Methods | NC-Props** form to introduce a nonconventional user model in a simulation.

Nonconventional models calculate the property of a single nonconventional component on each call rather than for a set of components of a mixture. Therefore you must assign a model for each required property of each component. You can also specify model option codes.

User Models for Nonconventional Properties

Model-Specific Parameters

Property	Model Name	Principal Sub. Name	Name	Elements	Type
ENTHALPY	ENTHUSR	ENTHLU	ENTUA	5	Unary
DENSITY	DNSTYUSR	DNSTYU	DNSUA	5	Unary

Using Component Attributes

A nonconventional component is characterized in terms of its component attributes. You can combine any number of component attribute types, each of which may have several elements, to characterize a component.

A user nonconventional model may use any of the built-in component attribute types, as well as any of a set of five user attribute types:

CAUSR1, CAUSR2, CAUSR3, CAUSR4, CAUSR5

Each of these types has up to 10 user-defined elements. A user model may assume any definitions of these attributes. The only restriction is that the attributes should represent intensive rather than extensive quantities.

You can define components as nonconventional (Type=Nonconventional) on the **Components | Specifications | Selection** sheet.

Component Attributes and Offset Array Names

Component Attribute	Offset Array Name
PROXANAL	PRXANL
ULTANAL	ULTANL
SULFANAL	SLFANL
GENANAL	GENANL
CAUSR1	CAUSR1
CAUSR2	CAUSR2

Component Attribute	Offset Array Name
CAUSR3	CAUSR3
CAUSR4	CAUSR4
CAUSR5	CAUSR5

Principal User Model Subroutines for Nonconventional Properties

For each user model, you must provide a principal subroutine (see User Models for Nonconventional Properties), that calculates and returns the desired physical properties. Since the principal subroutines are called directly by the appropriate physical property monitors, they have a fixed name and argument list structure. The principal subroutine can call any additional subroutine. There is no restriction on the argument lists of these additional subroutines.

Calling Sequence for Nonconventional Properties

```
SUBROUTINE ENTHLU (T, P, CAT, IDXNC, IRW, IIW, KCALC, KOP, NDS,
                  KDIAG, Q, DQ, KER)
SUBROUTINE DNSTYU (T, P, CAT, IDXNC, IRW, IIW, KCALC, KOP, NDS,
                  KDIAG, Q, DQ, KER)
```

Argument Descriptions: Nonconventional Prop. Subroutines

Variable	I/O [†]	Type	Dimension	Description
T	I	REAL*8	—	Temperature (K)
P	I	REAL*8	—	Pressure (N/m ²)
CAT	I	REAL*8	(1)	Nonconventional component attribute vector (see Accessing Component Attributes)
IDXNC	I	INTEGER	—	Nonconventional component index (see IDXNC and Accessing Component Attributes)
IRW	I	INTEGER	—	Offset for real work array (see Real and Integer Work Areas)
IIW	I	INTEGER	—	Offset for integer work array (see Real and Integer Work Areas)
KCALC	I	INTEGER	—	Calculation code: KCALC = 1, Calculate property only KCALC = 2, Calculate temperature derivative of property only KCALC = 3, Calculate property and temperature derivative
KOP	I	INTEGER	10	Integer vector containing up to 10 model option codes (see KOP)

Variable	I/O [†]	Type	Dimension	Description
NDS	I	INTEGER	—	Data set number of the model-specific parameters (see Parameter Retrieval)
KDIAG	I	INTEGER	—	Diagnostic level code for the control of warning and error messages (see KDIAG)
Q	O	REAL*8	—	Property of a single attributed component
DQ	O	REAL*8	—	Temperature derivative of the property of a single attributed component
KER	O	INTEGER	—	Error return code (not used)

[†] I = Input to subroutine, O = Output from subroutine

IDXNC

IDXNC is the index of the single nonconventional component for which calculations are to be performed. It is the sequence number of the component within the complete set of attributed components, both conventional and nonconventional, in the simulation.

Real and Integer Work Areas

To access the real and integer work areas reserved for a user physical property model:

- 1 Include labeled commons DMS_PPWORK and DMS_IPWORK. Include directives begin in column 1.

```
#include "dms_ppwork.cmn"
#include "dms_ipwork.cmn"
```

- 2 Include labeled common DMS_PLEX. Include directives begin in column 1.

```
#include "dms_plex.cmn"
```

- 3 Add the following declarations:

```
REAL*8 B(1)
EQUIVALENCE (B(1), IB(1))
```

The first element of the real work area that the user model can use is:

```
B(PPWORK_LPPWK + IRW + 1)
```

The length of this area is 20 elements.

The first element of the integer work area that the user model can use is:

```
IB(IPWORK_LIPWK + IIW + 1)
```

The length of this area is 20 elements. Both the integer and real work areas are local to the user subroutine, and their contents are not saved/retained from one call to the next.

KOP

You can use the 10-element option code vector, KOP, to provide additional calculational flexibility within the model. For example, multiple correlations within the same user model may be handled by branching to different

segments of a subroutine, or to different subroutines, depending on the option codes. You can assign specific integer values to the option codes on the **Methods | NC-Props** form. Option codes default to zero for all user models.

KDIAG

You can use KDIAG to control the printing of error, warning, and diagnostic information in the History File. The values for KDIAG correspond to the values for message levels described in the Help screens for Physical Properties on the **Setup | Specifications | Diagnostics** sheet or the **Block Options | Diagnostics** sheet of a unit operation model. If KDIAG=N, all messages at level N and below will be printed.

Range of Applicability

A user model should test for the range of applicability of the model, and provide a reasonable extrapolation beyond the range. A warning message should be written to the History File. Frequently, the reasonable extrapolation can be a linear extrapolation with a slope equal to that of the correlation at or near the appropriate bound. When the correlation bounds are violated or when any other condition occurs that may make the results questionable, an error or warning message should be written to the History File (see Chapter 4, Aspen Plus Error Handler).

Model-Specific Parameters for Nonconventional Properties

Each user model for nonconventional properties has a built-in model-specific unary parameter. The names of these parameters are listed in this section. You can access these parameters by:

- Including the labeled common DMS_PLEX in the model.
- Determining the offset of the parameter into the PLEX by calling the utility DMS_IFCMN (or DMS_IFCMNC) (see Chapter 4).

If additional model-specific parameters are required, you can introduce them by assigning names to them on the **Customize | User Parameters** form. These parameters are stored in the labeled common DMS_PLEX, and can be accessed by first determining the offset in the PLEX.

If parameters required by a user model are already known to Aspen Plus because of their association with built-in Aspen Plus models, but are not required by any models invoked by the user, you must include these parameters on the **Customize | User Parameters** form.

All binary user model parameters default to zero, and all unary parameters default to RGLOB_RMISS (from COMMON/DMS_RGLOB/). You can enter values of the model-specific parameters on the **Methods | Parameters** forms.

Naming Model-Specific Parameters

The name of the additional model-specific parameters should be chosen carefully to avoid conflict.

Accessing Component Attributes

Aspen Plus passes the nonconventional component attribute vector, CAT, to the user principal subroutine through its argument list. CAT stores the values of the component attributes for the complete set of attributed components for each stream, both conventional and nonconventional. You can retrieve a particular element of an attribute from the CAT vector using an index stored in the plex. The plex location is determined from the component index IDXNC and another plex offset of the attribute. Use the utility DMS_IFCMN or DMS_IFCMNC to obtain the offset. See the example below.

Example: Accessing Component Attributes

Retrieve the IELth element of the attribute CAUSR3 and assign it to scalar variable CATI. The following Fortran statements are used:

```
#include "dms_plex.cmn"
      DIMENSION CAT(1)
      INTEGER DMS_IFCMNC
      .
      .
      .
      LCAU3 = DMS_IFCMNC ('CAUSR3')
      I = IEL + IB(LCAU3 + IDXNC)
      CATI = CAT(I)
```

To retrieve values of model-specific parameters from the labeled common, DMS_PLEX, IDXNC index is again used. For example, the IELth element of the five-element parameter ENTUA can be obtained as follows:

```
#include "dms_plex.cmn"
      INTEGER DMS_IFCMNC
      REAL*8 B(1)
      EQUIVALENCE (B(1), IB(1))
      .
      .
      .
      LENTUA = DMS_IFCMNC('ENTUA')
      K = LENTUA + 5 * (IDXNC - 1)
      ENTUAI = B(K + IEL)
```


7 User Properties for Property Sets

You can define user properties for a Property Set by supplying the Fortran subroutines to calculate the property values. First, you define the property on the **Customize | User Properties | Specifications** sheet. Then you can reference the property on the **Prop-Set | Properties** sheet just as you reference built-in Aspen Plus properties.

Subroutine to Define a User Property

Calling Sequence for User Properties

```
SUBROUTINE subrname† (T, P, FV, FL, BETA, N, IDX, FLOW, Y, X, X1, X2, Z, NBOPST, KDIAG, KPDIAG, XPCTLV, IPHASE, NAME, PRPVAL, SF, S, ISUBS, CAT, FLOWS, NSUBS, IDXSUB, ITYPE)
```

[†] Subroutine name you entered on the **Customize | User Properties | Specifications** sheet.

Argument Descriptions for User Properties

Variable	I/O [†]	Type	Dimension	Description
T	I	REAL*8	—	Temperature (K)
P	I	REAL*8	—	Pressure (N/m ²)
FV	I	REAL*8	—	Molar vapor fraction
FL	I	REAL*8	—	Molar liquid fraction
BETA	I	REAL*8	—	Moles liquid 1/moles total liquid
N	I	INTEGER	—	Number of components present (see IDX)
IDX	I	INTEGER	N	Component index vector for Y, X, X1, X2, and Z (and PRPVAL for component properties) (see IDX)
FLOW	I	REAL*8	—	Total molar flow (kgmole/s) (see IDX)

Variable	I/O [†]	Type	Dimension	Description
Y	I	REAL*8	N	Mole fraction vector for vapor phase (see IDX)
X	I	REAL*8	N	Mole fraction vector for liquid phase (see IDX)
X1	I	REAL*8	N	Mole fraction vector for liquid phase 1 (see IDX)
X2	I	REAL*8	N	Mole fraction vector for liquid phase 2 (see IDX)
Z	I	REAL*8	N	Total (all phases combined) mole fraction vector (see IDX)
NBOPST	I	INTEGER	6	Physical Property option set vector (see NBOPST, KDIAG, and KPDIAG)
KDIAG	I	INTEGER	—	Simulation diagnostic level (Simulation level set globally on Setup Specifications Diagnostics sheet) (see NBOPST, KDIAG, and KPDIAG)
KPDIAG	I	INTEGER	—	Property diagnostic level (Physical Properties set globally on Setup Specifications Diagnostics sheet) (see NBOPST, KDIAG, and KPDIAG)
XPCTLV	I	REAL*8	—	Distillation curve liquid volume percent
IPHASE	I	INTEGER	—	Phase for which calculations are requested (Phase on Prop-Set Qualifiers sheet) (see Phase Codes) 1 = vapor, 2 = liquid, 3 = solid 4 = second liquid, 5 = total (all phases combined), 6 = first liquid
NAME	I	INTEGER	2	Property name stored as 2 integer words
PRPVAL	O	REAL*8	— or (N)	For mixture properties: one property value For pure component or partial properties: vector (of length N) of component property values (see IDX)
SF	I	REAL*8	—	Solid fraction (1.0-FV-FL)
S	I	REAL*8	N	Mole or mass fraction vector for solid phase (mole for substreams of type CISOLID, mass for substreams of type NC)
ISUBS	I	INTEGER	4	Substream for which calculations are requested (1)= Substream type (1-MIXED, 2-CISOLID, 3-NC) (2)= Substream number (3) and (4)= Substream ID stored as 2 integers (Substream on the Properties Prop-Sets Qualifiers sheet) If ISUBS(1)=0, calculations are requested for the total stream (Substream=ALL on the Prop-Set Qualifiers sheet)
CAT	I	REAL*8	(1)	Component attribute vector
FLows	I	REAL*8	(1)	Stream vector (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in FLows
IDXSUB	I	INTEGER	NSUBS	Location of substreams within FLows vector

Variable	I/O [†]	Type	Dimension	Description
ITYPE	I	INTEGER	NSUBS	Substream type vector for FLOWS (1-MIXED, 2-CISOLID, 3-NC)

[†] I = Input to subroutine, O = Output from subroutine

IDX

IDX is a vector containing component sequence numbers of the conventional components actually present in the order that they appear on the **Components | Specifications | Selection** sheet. For example, if only the first and third conventional components are present, then N=2 and IDX=(1,3).

The mole fraction vectors Y, X, X1, X2, and Z contain mole fractions only for components in the stream. In the above example X(2) would be the liquid mole fraction for the component listed third on the **Components | Specifications | Selection** sheet. For component properties, fill in PRPVAL only for the components actually present.

NBOPST, KDIAG, and KPDIAG

Your subroutine should use the variables NBOPST, KDIAG and KPDIAG only if it calls Aspen Plus physical property monitors or utility routines (see Chapter 3). These variables should be passed to the Aspen Plus subroutine.

Phase Codes

Aspen Plus calls the user subroutine for each phase you listed on the **Prop-Set | Qualifiers** sheet, provided that phase exists in the stream. The user subroutine must check for valid phase codes.

Passing Phase Fraction and Composition Information

The phase fraction and composition information that Aspen Plus passes depends on the phase you specified on the **Phase** field of the **Prop-Set | Qualifiers** sheet (IPHASE), and the stream flash option you specified on the Properties Advanced UserProperties Specifications sheet. If **Do Not Flash** is selected, Aspen Plus passes only the Z vector.

If *Always Flash* or *Do Not Flash for Total Phase Stream Properties* is selected, then:

When IPHASE =	Aspen Plus passes these arguments
1	Y
2	BETA, X, X1, X2
4	X2
5, Always Flash	FV, FL, BETA, Y, X, X1, X2

When IPHASE =	Aspen Plus passes these arguments
5, Do Not Flash for Total Phase Stream Properties	Z
6	X1

Component Order Independence

In some cases, you might want to create a user Property Set subroutine that is independent of the order in which you specified components on the **Components | Specifications | Selection** sheet. You can use the component alias and call the utility DMS_KFORM or DMS_KFORMC for this purpose (see Chapter 4).

For a component	Then
In an Aspen Plus databank	Use the component's databank alias
Not in a databank	Use the User-Defined Component Wizard from the Components Specifications Selection sheet to enter the component alias

8 User Stream Report

The Aspen Plus stream report interface allows you to access any or all of the streams in the simulation for writing user-defined stream reports or for interfacing to downstream programs.

Enter the user subroutine name on the **Setup | Report Options | Stream** sheet. Click on the **Supplementary Stream** button to create a Supplemental stream report. On the **Supplemental Stream Report** dialog box, click the **Subroutine** Button to specify the subroutine name.

Stream Report Subroutine

Calling Sequence for Stream Report Subroutine

SUBROUTINE subrname[†] (STVEC, NSTRM, NTOT, IDSTRM, IDBSOR, IDBSNK, IHEAD, LHEAD, ISUPNO, NOSAT, ISCTYP, NPTOT, IHEADS, PRPVAL, XTREF, XPREF, XPCLV, NSUBS, IDXSUB, ITYPE, LD)

[†] Subroutine name you entered on the **Supplemental Stream Report Subroutine** dialog box.

Argument List Descriptions for Stream Report Subroutine

Variable	I/O [†]	Type	Dimension	Description
STVEC	I	REAL*8	NTOT, NSTRM	Stream vectors (see Stream Classes)
NSTRM	I	INTEGER	—	Number of streams
NTOT	I	INTEGER	—	Number of stream variables
IDSTRM	I	INTEGER	2, NSTRM	Stream IDs
IDBSOR	I	INTEGER	2, NSTRM	Source block ID for each stream
IDBSNK	I	INTEGER	2, NSTRM	Sink block ID for each stream
IHEAD	I	INTEGER	LHEAD	Heading entered on the Supplemental Stream Report Subroutine dialog box.
LHEAD	I	INTEGER	—	Length of IHEAD
ISUPNO	I	INTEGER	—	Supplementary stream report number (Supplementary Report Number on the Supplemental Stream Report dialog box)

Variable	I/O [†]	Type	Dimension	Description
NOSAT	I	INTEGER	—	Number of stream attributes (0 for material streams; 1 for heat and work streams)
ISCTYP	I	INTEGER	—	Stream class type (1-material, 2-heat, 3-work)
NPTOT	I	INTEGER	—	Number of properties (see PRPVAL)
IHEADS	I	INTEGER	2, 6, NPTOT	Property headings Row 1: Name Row 2: Phase Row 3: Component Row 4: (Wet/Dry) or substream id ('wet' is not printed) Row 5: Units Row 6: More Units
PRPVAL	I	REAL*8	NPTOT, NSTRM	Property values (see PRPVAL)
XTREF	I	REAL*8	NPTOT	Reference temperatures (K)
XPREF	I	REAL*8	NPTOT	Reference pressures (N/m ²)
XPCLV	I	REAL*8	NPTOT	Liquid volume percents
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream types (1-MIXED, 2-CISOLID, 3-NC)
LD	I	INTEGER	—	Plex location of stream class bead

[†] I = Input to subroutine, O = Output from subroutine

Stream Classes

Aspen Plus calls the user subroutine once for each stream class. You can specify the streams that are passed to the user subroutine on the **Supplemental Stream Report Include Streams** dialog box (click the **Include Streams** button on the **Supplemental Stream Report** dialog box).

The stream structure for material streams is described in Appendix C. For information streams, the stream vector consists of a single value.

PRPVAL

The PRPVAL array contains all of the properties in all of the Property sets listed on the **Supplemental Stream Report Property Sets** dialog box (click the **Property Sets** button on the **Supplemental Stream Report** dialog box). For each property in each Property set, if there are any qualifiers, the array will contain values for all valid combinations of qualifiers. Qualifiers are specified on the **Prop-Set | Qualifiers** sheet. The qualifiers are varied in the following order, from fastest to slowest: Components, Phase, Temperature, Pressure, % Distilled, Units, and Water Basis. You can use the IHEADS array to locate values in the PRPVAL array. Use multiple Property sets to arrange the order of the values. Units for PRPVAL are the user's output units or the units in the UNITS qualifier.

NRPT

The Fortran unit number for the Aspen Plus report file is variable `USER_NRPT` in `COMMON /PPEXEC_USER/` (see Appendix A.) When writing to the Aspen Plus report file, subroutine `ZREP_RPTHDR` should be called before writing to ensure correct pagination (see Report Header Utility, Chapter 4.)

Component IDs

Component IDs are stored in `COMMON /DMS_PLEX/` in the `IDSCC` area for conventional components and in `COMMON /DMS_PLEX/` in the `IDSNCC` area for nonconventional components (see Appendix A).

9 User Blending Subroutines for Petroleum Properties

Aspen Plus provides mole average, weight average, and volume average blending rules for petroleum properties. If these are not sufficient, you can provide your own blending rules. To supply your own blending rule:

- 1 On the **Customize | User Properties | Specifications** sheet, specify select Assay Curve Property. In the **Blending Method** specification, select **Blending calculation provided in user subroutine BLDPPU**, then specify the blending option, if any.
- 2 Write a Fortran subroutine named BLDPPU to calculate the bulk property using your own blending rules. If you use different blending rules for different properties, use the blend option to branch to the appropriate sections inside your subroutine for blending different properties.

Petroleum Blending Subroutine

Calling Sequence for Blending Subroutine

```
SUBROUTINE BLDPPU (NCOMP, IDX, KODEXT, ITYPE, NOPT, IOPT, PP,  
                  VOL, WT, XMOL, SG, XMW, BLKVAL, WTF)
```

Argument List Descriptions for Blending Subroutine

Variable	I/O [†]	Type	Dimension	Description
NCOMP	I	INTEGER	—	Number of components present
IDX	I	INTEGER	NCOMP	Component index vector (see IDX)
KODEXT	I	INTEGER	—	Extrapolation code: 1 = Yes, 2 = No
ITYPE	I	INTEGER	—	Not used
NOPT	I	INTEGER	—	Not used
IOPT	I	INTEGER	—	Blending option you specified on the Properties Advanced UserProperties Specifications sheet
PP	I	REAL*8	NCOMP	Petroleum property values
VOL	I	REAL*8	NCOMP	Volume fraction

Variable	I/O [†]	Type	Dimension	Description
WT	I	REAL*8	NCOMP	Weight fraction
XMOL	I	REAL*8	NCOMP	Mole fraction
SG	I	REAL*8	NCOMP	Specific gravity
XMW	I	REAL*8	NCOMP	Molecular weight
BLKVAL	O	REAL*8	—	Petroleum property bulk value
WTF	I	REAL*8	NCOMP	Not used

[†] I = Input to subroutine, O = Output from subroutine

IDX

IDX is a vector containing component sequence numbers of the conventional components actually present in the order that they appear on the **Components | Specifications | Selection** sheet. For example, if only the first and third conventional components are present, then NCOMP=2 and IDX=(1,3).

10 User Subroutines for Sizing and Costing

The user unit operation model User2 can call Fortran subroutines that you supply to do sizing and costing calculations.

Sizing Model Subroutine

Calling Sequence for Sizing Model Subroutine

```
SUBROUTINE subrname†      (KMODE, NEQP, NINT, INT, NREAL, REAL,  
                             NIST, ICSIZ, RCSIZ, NRSLT, IRSLT, RSLT)
```

[†] Subroutine name you entered.

Argument List Descriptions for Sizing Model Subroutine

Variable	I/O [†]	Type	Dimension	Description
KMODE	I	INTEGER	—	Costing mode flag = 1: Uses user written costing model = 2: Uses user correlation
NEQP	I/O	INTEGER	—	Number of pieces of equipment. Aspen Plus sets this value to the integer missing value USER_IUMISS (from COMMON /PPEXEC_USER/) if you do not specify. If it is not set, it should be calculated by the sizing model.
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
NIST	I	INTEGER	—	Number of values retrieved from flowsheet reference

Variable	I/O[†]	Type	Dimension	Description
ICSIZ	I	INTEGER	NIST	Array of reference codes. See ICSIZ.
RCSIZ	I	REAL*8	NIST	Array of flowsheet reference values. This array parallels the ICSIZ array, so an element of RCSIZ contains the value for the code in the same element of ICSIZ.
NRSLT	I	INTEGER	—	Number of size results you specified
IRSLT	I	INTEGER	NRSLT	Array of size-result codes. See IRSLT.
RSLT	O	REAL*8	NRSLT	Array of size results you entered. This array parallels the RSLT array, identifying results of the types specified in IRSLT.

[†] I = Input to subroutine, O = Output from subroutine

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number of values. You can initialize these parameters by assigning values. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

ICSIZ

The following values in array ICSIZ correspond to the specifications listed on the User2 reference form:

10 PIN	15 VAP-FLOW-OUT	51 PREF	90 NFEEDS
11 POUT	15 VAP-FLOW	52 HC-RATIO	91 NSTAGE
11 PRES	16 LIQ-FLOW	53 MEFF	92 SURF-TEN
12 TIN	17 VAP-DENSITY	54 COMPR-TYPE	93 VISCOSITY
13 TOUT	18 LIQ-DENSITY	55 HP	94 REL-VOL
13 TEMP	19 DUTY	56 ZIN	95 LIQ-DIFF
14 VAP-FLOW-IN	50 AREA	57 ZOUT	

IRSLT

The following integer values in array IRSLT correspond to the results codes entered in RSLT:

1 PRES	6 AREA	10 DIAM	14 FLOW-PARAM
2 TEMP	7 HP	11 TT-LENGTH	15 VOLUME
3 VAP-FLOW	8 HEAD	12 THICKNESS	16 CAPACITY
4 LIQ-FLOW	9 WEIGHT	13 VOL-FLOW	17 NTRAY
5 DUTY			

Costing Model Subroutine

Calling Sequence for Costing Model Subroutine

SUBROUTINE subrname[†] (KMODE, NEQP, FMCN, CSTAJ, NIST, ICSIZ, RCSIZ, NRSLT, IRSLT, RSLT, TBCOST, TCOST

[†] Subroutine name you entered.

Argument List for Costing Model Subroutine

Variable	I/O [†]	Type	Dimension	Description
KMODE	I	INTEGER	—	Costing mode flag. = 1: Uses user written costing model = 2: Uses user correlation
NEQP	I	INTEGER	—	Number of pieces of equipment
FMCN	I/O	REAL*8	—	Material of construction factor. May be specified or calculated by the model.
CSTAJ	I	REAL*8	—	Costing adjustment factor you specified
NIST	I	INTEGER	—	Number of values retrieved from flowsheet reference
ICSIZ	I	INTEGER	NIST	Array of reference data codes. See sizing model argument list description for description of values.
RCSIZ	I	REAL*8	NIST	Array of values retrieved from flowsheet reference.
NRSLT	I	INTEGER	—	Number of user specified size results
IRSLT	I	INTEGER	NRSLT	Array of user specified size-results codes. See sizing model argument list description for description of values.
RSLT	I	REAL*8	NRSLT	Array of user specified size-results
TBCOST	O	REAL*8	—	Total calculated base cost per piece of equipment
TCOST	O	REAL*8	—	Total calculated adjusted cost per piece of equipment. (TCOST = TBCOST * CSTAJ * FMCN)

[†] I = Input to subroutine, O = Output from subroutine

11 User Kinetics Subroutines

You can supply a kinetics subroutine to calculate reaction rates for the following models.

Model Name	Description
RPlug	Kinetic reactor model
RBatch	Kinetic reactor model
RCSTR	Kinetic reactor model
RadFrac	Staged separation model
Pres-Relief	Pressure relief system

For RadFrac, you can supply the user kinetics subroutine in either USER reaction type or REAC-DIST reaction type.

For descriptions of the kinetics and other subroutines for reactions of type Segment-based, Step-growth, or Free-radical, see the chapters of Aspen Polymers User Guide, Volume 1 devoted to these models. For the Gel Effect subroutine for Emulsion reactions, see the Gel Effect section in the chapter for Free-Radical reactions.

Kinetics Subroutine for USER Reaction Type

You can use this user kinetics subroutine for the reactor unit operation models (RPlug, RBatch, RCSTR), the staged separation model (RadFrac), and the Pressure Relief system (Pres-Relief).

For the reactor models and the pressure relief system, the kinetics subroutine calculates the rate of generation for each component in each substream. If solids participate in the reactions, the kinetics subroutine also accounts for changes in the outlet stream particle size distribution, and in the component attribute values. For example, if coal is treated as a single non-conventional component, its attributes, as represented by its ultimate and proximate analysis, must change at a rate consistent with the reactions.

For RadFrac, the kinetics subroutine calculates the rate of generation for each component on a given stage. It should also calculate the individual reaction rates in each phase if you use this routine for rate-based calculations.

On the Reactions Reactions form select the USER type to specify the reaction stoichiometry and the associated user-supplied kinetics subroutine name.

In addition to the parameters passed to your subroutine through the argument list, there are common blocks you can use to access parameters specific to a unit operation model.

Calling Sequence for User Kinetics Subroutine, Type USER

```
SUBROUTINE subrname† (SOUT, NSUBS, IDXSUB, ITYPE, NINT, INT,
                        NREAL, REAL, IDS, NPO, NBOPST, NIWORK,
                        IWORK, NWORK, WORK, NC, NR, STOIC, RATES,
                        FLUXM, FLUXS, XCURR, NTCAT, RATCAT,
                        NTSSAT, RATSSA, KCALL, KFAIL, KFLASH,
                        NCOMP, IDX, Y, X, X1, X2, NRALL, RATALL,
                        NUSERV, USERV, NINTR, INTR, NREALR,
                        REALR, NIWR, IWR, NWR, WR, NRL, RATEL,
                        NRV, RATEV)
```

[†] Subroutine name you entered on the **Reactions | Subroutine** sheet.

Argument List for User Kinetics Subroutine, Type USER

Variable	I/O/W [†]	Type	Dimension	Description
SOUT	I	REAL*8	1	Stream vector (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1 = MIXED 2 = CISOLID 3 = NC)

Variable	I/O/W [†]	Type	Dimension	Description
NINT	I	INTEGER	—	Number of integer parameters, from unit operation block User Subroutine form (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters, from unit operation block User Subroutine form (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters, from unit operation block User Subroutine form (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters, from unit operation block User Subroutine form (see Integer and Real Parameters)
IDS	I	INTEGER	2,4	(*,1) - Block ID as Hollerith string (*,2)-(*,3) - Used by Aspen Plus (1,4) - Type of the unit operation block (see Calling Model Type)
NPO	I	INTEGER	—	Number of property option sets
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector, from unit operation block User Subroutine form (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector, from unit operation block User Subroutine form (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector, from unit operation block User Subroutine form (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector, from unit operation block User Subroutine form (see Local Work Arrays)
NC	I	INTEGER	—	Total number of components defined on the Components Specifications Selection sheet
NR	I	INTEGER	—	Number of User kinetic reactions
STOIC	I	REAL*8	NC, NSUBS, NR	Array of stoichiometric coefficients (see STOIC)
RATES	O	REAL*8	(1)	Component reaction rates from User kinetic reactions (see Reaction Rates)
FLUXM	O	REAL*8	(1)	Component fluxes, mixed-solid (see Component Fluxes in RPlug)
FLUXS	O	REAL*8	(1)	Component fluxes, solid-mixed (RPlug) (see Component Fluxes in RPlug)
XCURR	I	REAL*8	—	RPlug: Current reactor location(m) RBatch, Pres-Relief: Current time(s) RCSTR: Current iteration number RadFrac: Current stage number
NTCAT	I	INTEGER	—	Dimension of RATCAT
RATCAT	O	REAL*8	NTCAT	Rates of component attributes (see Component Attributes and Substream PSD)

Variable	I/O/W [†]	Type	Dimension	Description
NTSSAT	I	INTEGER	—	Dimension of RATSSA
RATSSA	O	REAL*8	NTSSAT	Rates of substream PSD weight fractions (RCSTR, RPlug) (see Component Attributes and Substream PSD)
KCALL	I	INTEGER	—	Call code (RPlug, RBatch, RCSTR, Pres-Relief) 0 = First call and calculate rates 1 = Calculate rates 11 = Output point. Print to history file 12 = Final call before exiting 13 = Report pass. Print to report file
KFAIL	O	INTEGER	—	Fail Code (RPlug, RBatch, RCSTR, Pres-Relief) 0 = Continue calculations 1 = Model is failing for current conditions 99 = Stop calculations
KFLASH	I	INTEGER	—	Stream flash flag (RPlug, RBatch, RCSTR, Pres-Relief) 0 = Stream has not been flashed 1 = Stream has been flashed
NCOMP	I	INTEGER	—	Number of components present
IDX	I	INTEGER	NCOMP	Component sequence numbers
Y	I	REAL*8	NCOMP	Vapor mole fractions
X	I	REAL*8	NCOMP	Liquid mole fractions
X1	I	REAL*8	NCOMP	Liquid 1 mole fractions
X2	I	REAL*8	NCOMP	Liquid 2 mole fractions
NRALL	I	INTEGER	—	Total number of kinetic Powerlaw and LHHW reactions in the block (RPlug, RBatch, RCSTR, Pres-Relief)
RATALL	I	REAL*8	NRALL	Individual reaction rates of kinetic Powerlaw and LHHW reactions (RPlug, RBatch, RCSTR, Pres-Relief) (see Reaction Rates)
NUSERV	I	INTEGER	—	Number of user variables (RPlug, RBatch) (from User Subroutine User Variables sheet)
USERV	I/O	REAL*8	NUSERV	Vector of user variables (RPlug, RBatch) (from User Subroutine User Variables sheet; see User Variables)
NINTR	I	INTEGER	—	Number of integer parameters (from Reactions Subroutine sheet) (see Integer and Real Parameters)
INTR	I/O	INTEGER	NINTR	Vector of integer parameters (from Reactions Subroutine sheet) (see Integer and Real Parameters)
NREALR	I	INTEGER	—	Number of real parameters (from Reactions Subroutine sheet) (see Integer and Real Parameters)

Variable	I/O/W [†]	Type	Dimension	Description
REALR	I/O	REAL*8	NREALR	Vector of real parameters (from Reactions Subroutine sheet) (see Integer and Real Parameters)
NIWR	I	INTEGER	—	Length of integer work array (from Reactions Subroutine sheet) (see Local Work Arrays)
IWR	W	INTEGER	NIWR	Integer work array (from Reactions Subroutine sheet) (see Local Work Arrays)
NWR	I	INTEGER	—	Length of real work array (from Reactions Subroutine sheet) (see Local Work Arrays)
WR	W	REAL*8	NWR	Real work array (from Reactions Subroutine sheet) (see Local Work Arrays)
NRL	I	INTEGER	—	Number of liquid phase user kinetic reactions (RadFrac)
RATEL	O	REAL*8	NRL	Individual reaction rates in the liquid phase (kgmole/s) (for Rate-Based Distillation)
NRV	I	INTEGER	—	Number of vapor phase user kinetic reactions (RadFrac)
RATEV	O	REAL*8	NRV	Individual reaction rates in the vapor phase (kgmole/s) (for Rate-Based Distillation)

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

Integer and Real Parameters

You can use two sets of integer and real retention parameters:

- Parameters specific to the unit operation block (on the **User Subroutine** form).
- Parameters specific to the set of reactions (on the **Reactions | Subroutine** sheet).

You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed.

Local Work Arrays

You can use local work arrays by specifying the array length on the:

- User Subroutine** form for the unit operation block.

- **Reactions | Subroutine** sheet.

Aspen Plus does not retain these arrays from one call to the next.

Calling Model Type

The meanings of some arguments to the user kinetics subroutine depend on the type of unit operation model which called it. Element IDS(1,4) specifies this model type as a Hollerith string. The possible values are:

IDS(1,4) Value	Calling Model Type
RBAT	RBatch
RCST	RCSTR
RPLU	RPlug
PRES	Pres-Relief
RADF	RadFrac

Example:

```

      INTEGER MODELS(5)
      DATA MODELS /4HRBAT,4HRCST,4HRPLU,4HPRES,4HRADF/
C ...
      IF (IDS(1,4) .EQ. MODELS(1)) THEN
C Code for RBatch here
      ELSE IF (IDS(1,4) .EQ. MODELS(2)) THEN
C Code for RCSTR here
C ...

```

STOIC

For RadFrac, the STOIC array is sorted such that the coefficients for the liquid phase reactions appear after all the vapor-phase reactions. The sort results do not depend on the order in which you entered the reactions on the

Reactions | Subroutine sheet.

Reaction Rates

The component reaction rates must be calculated for each component. These are rates of change and should be negative for a component that is consumed and positive for a component that is generated. When there are multiple substreams, the structure of the RATES vector is the concatenation of the rates for all components in the first substream, rates for all components in the second substream, etc. Substream structure is discussed in Appendix C.

The units of the reaction rates are:

- For RBatch, RCSTR, and Pres-Relief.
 Conventional components: $(\text{kgmole}/\text{m}^3\text{-s}) (\text{m}^3) = \text{kgmole}/\text{s}$
 Nonconventional components: $(\text{kg}/\text{m}^3\text{-s}) (\text{m}^3) = \text{kg}/\text{s}$
 (rates per unit volume are multiplied by the volume occupied by the reacting phase)

- For Rplug.
Conventional components: $(\text{kgmole}/\text{m}^3\text{-s}) (\text{m}^2) = \text{kgmole}/\text{m-s}$
Nonconventional components: $(\text{kg}/\text{m}^3\text{-s}) (\text{m}^2) = \text{kg}/\text{m-s}$
(rates per unit volume are multiplied by the cross-sectional area covered by the reacting phase)
- For RadFrac.
Conventional components: (kgmole/s)

Component Attributes and Substream PSD

If solids participate in reactions, you need to account for changes in component attribute values and particle size distribution (PSD) of a substream. You can do one of the following:

- Update the attribute and PSD values in the stream vector SOUT in the user kinetics routine each time it gets called.
- Supply rates of change for the attributes and PSD weight fractions, in units of attribute/s and 1/s, respectively. In this case the reactor will converge on the attributes and PSD and will update the stream vector with their converged/current values each time it calls the user kinetics subroutine. To initiate this calculation for component attributes and/or PSD, use the **User Subroutine | Kinetics** sheet for RPlug and RBatch and use the **Setup | PSD** and **Setup | Component Attr** sheets for RCSTR. The structure of the component attribute rates vector (RATCAT) is: rates for all attributed components in the first substream, then the second substream, and so on. The structure for PSD rates vector (RATSSA) is rates for all PSD weight fractions in the first substream, then the second substream and so on.

User Variables

User variables provide a convenient method to access intermediate variables calculated within user subroutines.

As an example, consider a user kinetics routine that calculates rates for the following reactions:

- 1 $A + B \rightarrow C$
- 2 $B + 2D \rightarrow E$
- 3 $B \rightarrow F$

The subroutine returns the net reaction rates for components A through F in the RATES vector. However, for diagnostic purposes, it may be more convenient to view the rates of individual reactions 1 through 3. These types of intermediate calculations can be loaded into the USERV array.

The user variables are identified with user-specified labels. These labels are specified on the reactor model (RBatch or RPlug) **User Subroutine | User Variables** sheet.

The user variable results are tabulated in the reactor model's **Profiles | User Variables** sheet. User variables can also be tabulated in the simulation history file by setting the **Simulation level** higher than 5 on the **Block Options | Diagnostics** sheet.

Aspen Plus does not perform unit conversion on the user variables. User variables can be stored by any and all of the user subroutines called by a reactor model.

Component Fluxes in RPlug

FLUXM and FLUXS are used to take into account the effect of mass exchange on the substream enthalpy balance in RPlug when multiple substreams exist in your simulation and you have specified that the substream temperatures may be different (on the **RPlug | Setup | Specifications** sheet).

Reactions can occur within each substream and can use reactants from, or add products to, any substream. Each reaction must be assigned to a specific substream so that the temperature used to determine the reaction rate is the temperature of that substream, and the reaction heat effect is absorbed by that substream.

Both FLUXM and FLUXS give the flux (molar flow rate per unit volume) of components between substreams due to reactions specified for another substream, but with a different sign and a different basis for the volume:

- FLUXM is the flux, at mixed substream temperature, from mixed substream to solid substream.
- FLUXS is the flux, at solid substream temperature, from solid substream to mixed substream.

The structure and units of both FLUXM and FLUXS are identical to those of the RATES vector.

COMMON RPLG_RPLUGI

COMMON block RPLG_RPLUGI contains integer configuration parameters for the RPlug block. To use RPLG_RPLUGI, include the following directive (in column 1) of your subroutine:

```
#include "rplg_rplugi.cmn"
```

The following table describes the variables in RPLG_RPLUGI.

Variable	Type	Description
RPLUGI_IRPLUG	INTEGER	RPlug reactor type (from the RPlug Setup Specifications sheet; see below for explanation of values)
RPLUGI_NTUBE	INTEGER	Number of reactor tubes

Values of RPLUGI_IRPLUG have the following meanings:

- 1 = Reactor with Specified Temperature
- 2 = Adiabatic Reactor
- 3 = Reactor with Co-current Coolant
- 4 = Reactor with Counter-current Coolant
- 5 = Adiabatic Reactor (with substreams at different temperatures)
- 6 = Reactor with Co-current Coolant (with substreams at different temperatures)
- 7 = Reactor with Counter-current Coolant (with substreams at different temperatures)
- 8 = Reactor with Constant Coolant Temperature

9 = Reactor with Constant Coolant Temperature (with substreams at different temperatures)

COMMON RPLG_RPLUGR

COMMON block RPLG_RPLUGR contains real configuration parameters for the RPlug block. To use RPLG_RPLUGR, include the following directive (in column 1) of your subroutine:

```
#include "rplg_rplugr.cmn"
```

The following table describes the variables in RPLG_RPLUGR.

Variable	Type	Description
RPLUGR_UXLONG	REAL*8	Reactor length (m)
RPLUGR_UDIAM	REAL*8	Reactor (or tube) diameter (m)
RPLUGR_AXPOS	REAL*8	Current axial position (m)
RPLUGR_TCOOL	REAL*8	Coolant temperature (from Setup Specifications sheet) (K)
RPLUGR_CATWT	REAL*8	Total catalyst mass (kg)
RPLUGR_CAT_RHO	REAL*8	Catalyst mass density (kg/m ³)
RPLUGR_BED_VOID	REAL*8	Reactor bed void fraction (unitless)

COMMON RBTC_RBATI

COMMON block RBTC_RBATI contains integer configuration parameters for the RBatch block. To use RBTC_RBATI, include the following directive (in column 1) of your subroutine:

```
#include "rbtc_rbatl.cmn"
```

The following table describes the variables in RBTC_RBATI.

Variable	Type	Description
RBATI_IRBAT	INTEGER	RBatch reactor type (from RBatch Setup Specification sheet) 1 = Constant Temperature 2 = Temperature Profile 3 = Constant Heat Duty 4 = Constant Coolant Temperature 5 = Heat Duty Profile 6 = Heat Transfer User Subroutine

COMMON RBTC_RBATR

COMMON block RBTC_RBATR contains real configuration parameters for the RBatch block. To use RBTC_RBATR, include the following directive (in column 1) of your subroutine:

```
#include "rbtc_rbatr.cmn"
```

The following table describes the variables in RBTC_RBATR.

Variable	Type	Description
----------	------	-------------

RBATR_VOLRB	REAL*8	RBatch reactor volume (m ³)
-------------	--------	---

COMMON RCST_RCSTRI

COMMON block RCST_RCSTRI contains integer configuration parameters for the RCSTR block. To use RCST_RCSTRI, include the following directive (in column 1) of your subroutine:

```
#include "rcst_rcstri.cmn"
```

The following table describes the variables in RCST_RCSTRI.

Variable	Type	Description
RCSTRI_IRCSTR	INTEGER	RCSTR reactor type: 1 = Temperature specified, 2 = Heat duty specified

COMMON RXN_RCSTRR

COMMON block RXN_RCSTRR contains real configuration parameters for the RCSTR block. To use RXN_RCSTRR, include the following directive (in column 1) of your subroutine:

```
#include "rxn_rcstrr.cmn"
```

The following table describes the variables in RXN_RCSTRR.

Variable	Type	Description
RCSTRR_VOLRC	REAL*8	RCSTR reactor volume
RCSTRR_VFRR	REAL*8	RCSTR volume fraction of phase specified for holdup (Setup Specifications sheet)
RCSTRR_CATWT	REAL*8	Total catalyst mass (kg)
RCSTRR_CAT_RHO	REAL*8	Catalyst mass density (kg/m ³)
RCSTRR_BED_VOI	REAL*8	Bed voidage (unitless)

COMMON PRSR_PRESRI

COMMON block PRSR_PRESRI contains integer configuration parameters for the Pres-Relief block. To use PRSR_PRESRI, include the following directive (in column 1) of your subroutine:

```
#include "prsr_presri.cmn"
```

The following table describes the variables in PRSR_PRESRI.

Variable	Type	Description
PRESRI_ISCNAR	INTEGER	Pressure relief scenario (from FlowsheetingOptions Pres-Relief Setup sheet) 1 = Dynamic run with vessel engulfed by fire 2 = Dynamic run with specified heat flux into vessel 6 = Steady state flow rating of relief system 7 = Steady state flow rating of relief valve

COMMON PRSR_PRESRR

COMMON block PRSR_PRESRR contains real configuration parameters for the Pres-Relief block. To use PRSR_PRESRR, include the following directive (in column 1) of your subroutine:

```
#include "prsr_presrr.cmn"
```

The following table describes the variables in PRSR_PRESRR.

Variable	Type	Description
PRESRR_VOLPR	REAL*8	Vessel volume (m ³)

COMMON RXN_DISTI

COMMON block RXN_DISTI contains integer configuration parameters for reactive distillation calculations. To use RXN_DISTI, include the following directive (in column 1) of your subroutine:

```
#include "rxn_disti.cmn"
```

The following table describes the variables in RXN_DISTI.

Variable	Type	Description
DISTI_IHLBAS	INTEGER	Basis for liquid holdup specification 1 = Volume 2 = Mass 3 = Mole
DISTI_IHVBAS	INTEGER	Basis for vapor holdup specification 1 = Volume 2 = Mass 3 = Mole

COMMON RXN_DISTR

COMMON block RXN_DISTR contains real configuration parameters for reactive distillation calculations. To use RXN_DISTR, include the following directive (in column 1) of your subroutine:

```
#include "rxn_distr.cmn"
```

The following table describes the variables in RXN_DISTR.

Variable	Type	Description
DISTR_TIMLIQ	REAL*8	Liquid residence time in stage or segment
DISTR_TIMVAP	REAL*8	Vapor residence time in stage or segment
DISTR_HLDLIQ	REAL*8	Liquid holdup in stage or segment (units depend on DISTI_IHLBAS, see below)
DISTR_HLDVAP	REAL*8	Vapor holdup in stage or segment (units depend on DISTI_IHVBAS, see below)
DISTI_IHLBAS		Units for DISTR_HLDLIQ and DISTR_HLDVAP
1		m ³
2		kg
3		kgmole

COMMON RXN_RPROPS

COMMON block RXN_RPROPS contains real property values, such as temperature and pressure, for the reaction calculations. To use RXN_RPROPS, include the following directive (in column 1) of your subroutine:

```
#include "rxn_rprops.cmn"
```

The following table describes the variables in RXN_RPROPS.

Variable	Type	Description
RPROPS_UTEMP	REAL*8	Reactor/stage temperature (K)
RPROPS_UPRES	REAL*8	Reactor/stage pressure (N/m ²)
RPROPS_UVFRAC	REAL*8	Molar vapor fraction in the reactor/stage
RPROPS_UBETA	REAL*8	Liquid 1/Total liquid molar ratio in the reactor/stage
RPROPS_UVVAP	REAL*8	For RBatch and RCSTR: Volume occupied by the vapor phase in the reactor (m ³) For RPlug: Cross-sectional area covered by the vapor phase in the reactor (m ²) For RadFrac: Vapor volume holdup in the stage [†] (m ³)
RPROPS_UVLIQ	REAL*8	For RBatch and RCSTR: Volume occupied by the liquid phase in the reactor (m ³) For RPlug: Cross-sectional area covered by the liquid phase in the reactor (m ²) For RadFrac: Liquid volume holdup in the stage [†] (m ³)
RPROPS_UVLIQS	REAL*8	For RBatch and RCSTR: Volume occupied by the liquid and solid phases in the reactor (m ³) For RPlug: Cross-sectional area covered by the liquid and solid phases in the reactor (m ²)

[†] Use only if you have specified the holdup for the stage on a volume basis.

User Kinetics Subroutine for REAC-DIST Reaction Type

You can use this user kinetics subroutine for the staged separation model RadFrac. The user kinetics subroutine calculates the generation rate (moles per unit time) for each component on a given stage. Use the **Reactions** form and select the REAC-DIST type to specify the reaction chemistry and the associated user-supplied kinetics subroutine name.

If your routine is used for rate-based calculations, it should use TLIQ and TVAP (rather than T) as the temperatures for each phase. It also must calculate the individual rates of each reaction in each phase.

Calling Sequence: User Kinetics Subroutine, Type REAC-DIST

SUBROUTINE subrname[†] (N, NCOMP, NR, NRL, NRV, T, TLIQ, TVAP, P, PHFRAC, F, X, Y, IDX, NBOPST, KDIAG, STOIC, IHLBAS, HLDLIQ, TIMLIQ, IHVBAS, HLDVAP, TIMVAP, NINT, INT, NREAL, REAL, RATES, RATEL, RATEV, NINTB, INTB, NREALB, REALB, NIWORK, IWORK, NWORK, WORK)

[†] Subroutine name you entered on the **Reactions | Subroutine** sheet.

Argument List: User Kinetics Subroutine, Type REAC-DIST

Variable	I/O [†]	Type	Dimension	Description
N	I	INTEGER	—	Stage number
NCOMP	I	INTEGER	—	Number of components present
NR	I	INTEGER	—	Total number of kinetic reactions
NRL	I	INTEGER	3	NRL (1) - Number of overall liquid reactions NRL (2) - Number of liquid1 reactions NRL (3) - Number of liquid2 reactions
NRV	I	INTEGER	—	Number of vapor phase kinetic reactions
T	I	REAL*8	—	Stage temperature (K)
TLIQ	I	REAL*8	—	Liquid temperature on stage (K)
TVAP	I	REAL*8	—	Vapor temperature on stage (K)
P	I	REAL*8	—	Stage pressure (N/m ²)
PHFRAC	I	REAL*8	3	Phase fraction (1) Vapor fraction (2) Liquid1 fraction (RadFrac, 3-phase) (3) Liquid2 fraction (RadFrac, 3-phase)
F	I	REAL*8	—	Total flow on stage or segment (Vapor + Liquid) (kgmole/s)
Y	I	REAL*8	NCOMP	Vapor mole fraction
IDX	I	INTEGER	NCOMP	Component sequence number
NBOPST	I	INTEGER	6	Property option set (see NBOPST)
KDIAG	I	INTEGER	—	Local diagnostic level
STOIC	I	REAL*8	NCOMP, NR	Reaction stoichiometry (see STOIC)
IHLBAS	I	INTEGER	—	Basis for liquid holdup specification 1: volume, 2: mass, 3: mole
HLDLIQ	I	REAL*8	—	Liquid holdup IHLBAS Units 1 m ³ 2 kg 3 kmol
TIMLIQ	I	REAL*8	—	Liquid residence time (s)
IHVBAS	I	INTEGER	—	Basis for vapor holdup specification 1: volume, 2: mass, 3: mole

[†]I = Input to subroutine, O = Output from subroutine, W = Workspace

Continued

Variable	I/O [†]	Type	Dimension	Description
HLDVAP	I	REAL*8	—	Vapor holdup IHVBAS Units 1 m ³ 2 kg 3 kmol
TIMVAP	I	REAL*8	—	Vapor residence time (s)
NINT	I	INTEGER	—	Number of integer parameters (from Reactions Subroutine sheet)
INT	I/O	INTEGER	NINT	Vector of integer parameters (from Reactions Subroutine sheet)
NREAL	I	INTEGER	—	Number of real parameters (from Reactions Subroutine sheet)
REAL	I/O	INTEGER	NREAL	Vector of real parameters (from Reactions Subroutine sheet)
RATES	O	REAL*8	NCOMP	Component reaction rates (kgmole/s)
RATEL	O	REAL*8	NRL	Individual reaction rates in the liquid phase (kgmole/s) (for Rate-Based Distillation)
RATEV	O	REAL*8	NRV	Individual reaction rates in the vapor phase (kgmole/s) (for Rate-Based Distillation)
NINTB	I	INTEGER	—	Number of integer parameters (from unit operation block User Subroutine form)
INTB	I/O	INTEGER	NINTB	Vector of integer parameters (from unit operation block User Subroutine form)
NREALB	I	INTEGER	—	Number of real parameters (from unit operation block User Subroutine form)
REALB	I/O	INTEGER	NREALB	Vector of real parameters (from unit operation block User Subroutine form)
NIWORK	I	INTEGER	—	Length of integer work vector (from unit operation block User Subroutine form)
IWORK	W	INTEGER	NIWORK	Integer work vector
NWORK	I	INTEGER	—	Length of real work vector (from unit operation block User Subroutine form)
WORK	W	INTEGER	NWORK	Real work vector

[†]I = Input to subroutine, O = Output from subroutine, W = Workspace

NBOPST

When calling FLSH_FLASH or a physical property monitor, NBOPST should be passed.

STOIC

STOIC is sorted such that the coefficients for liquid phase reactions appear after all vapor reactions regardless of the order in which you entered the reactions on the **Reactions | Subroutine** sheet.

12 User Pressure Drop and Holdup Subroutines for RPlug

You can supply a user pressure drop subroutine for the unit operation model RPlug. The user pressure drop subroutine calculates the stream pressure or pressure drop for both the process and coolant streams at any point along the reactor. You can specify the user subroutine on the **RPlug | User Subroutine | Pressure Drop** sheet.

You can supply a user holdup subroutine for the unit operation model RPlug. The user holdup subroutine calculates the reactor holdup at any point along the reactor. You can specify the user subroutine on the **RPlug | User Subroutine | Holdup** sheet.

RPlug Pressure Drop Subroutine

Calling Sequence for RPlug Pressure Drop Subroutine

SUBROUTINE subrname[†] (SIN, SOUT, SINC, SOUTC, NSUBS, IDXSUB, ITYPE, NINT, INT, NREAL, REAL, IDS, NPO, NBOPST, NIWORK, IWORK, NWORK, WORK, PRESP, PRESC, Z, XLONG, DIAM, ANGLE, ELEV, ROUGH, AFRAC, NTUBE, VOID, DPART, SPHER, NUSERV, USERV)

[†] Subroutine name you entered on the **RPlug | User Subroutine | Pressure Drop** sheet.

Argument List for RPlug Pressure Drop Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SIN	I	REAL*8	(1)	Inlet process stream vector (see Appendix C)
SOUT	I/O	REAL*8	(1)	Process stream vector at current reactor location Z (see Appendix C)
SINC	I	REAL*8	(1)	Inlet coolant stream vector (see Appendix C)
SOUTC	I/O	REAL*8	(1)	Coolant stream vector at current reactor location Z (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block IDs (* , 1) - Block ID (* , 2) to (* , 5) - Used by Aspen Plus (* , 6) - User pressure drop subroutine name (* , 7) - User heat transfer subroutine name (* , 8) - User holdup subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)

Variable	I/O/W [†]	Type	Dimension	Description
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector
PRESP	O	REAL*8	—	Process stream pressure (N/m ²) or pressure drop (dP/dZ) (N/m ² /m) (see Pressure)
PRESC	O	REAL*8	—	Coolant stream pressure (N/m ²) or pressure drop (dP/dZ) (N/m ² /m) (see Pressure)
Z	I	REAL*8	—	Current reactor location (m)
XLONG	I	REAL*8	—	Reactor Length, specified on the RPlug Setup Configuration sheet (m)
DIAM	I	REAL*8	—	Reactor (or tube) diameter, specified on the RPlug Setup Specification sheet (m)
ANGLE	I	REAL*8	—	Angle of inclination (radians) from the RPlug Setup Configuration sheet
ELEV	I	REAL*8	—	Elevation change from feed to discharge (m) from the RPlug Setup Configuration sheet
ROUGH	I	REAL*8	—	Roughness (m) from the RPlug Setup Pressure sheet
AFRAC	I	REAL*8	—	Fraction of cross-sectional area occupied by the condensed phases
NTUBE	I	REAL*8	—	Number of tubes from the RPlug Setup Configuration sheet
VOID	I	REAL*8	—	Void fraction from the RPlug Setup Catalyst sheet
DPART	I	REAL*8	—	Particle diameter from the RPlug Setup Catalyst sheet (m)
SPHER	I	REAL*8	—	Particle sphericity from the RPlug Setup Catalyst sheet
NUSERV	I	INTEGER	—	Number of user variables from the RPlug User Subroutine User Variables sheet
USERV	I/O	REAL*8	NUSERV	Vector of user variables from the RPlug User Subroutine User Variables sheet (See User Variables)

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

RPlug Holdup Subroutine

Calling Sequence for RPlug Holdup Subroutine

```
SUBROUTINE subrname†      (SOUT, NSUBS, IDXSUB, ITYPE, NINT,
                           INT, NREAL, REAL, IDS, NPO,
                           NBOPST, NIWORK, IWORK, NWORK, WORK,
                           XLONG, DIAM, VOID, THETA, Z, LAMBDA,
                           SIGMA, VISL, VISV, DENL, DENV,
                           VELMIX, KFLASH, HOLDUP)
```

[†] Subroutine name you entered on the **RPlug | User Subroutine | Holdup** sheet.

Argument List for RPlug Holdup Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SOUT	I/O	REAL*8	(1)	Process stream vector at current reactor location Z (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1: MIXED, 2: CISOLID, 3: NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block IDs (* , 1) - Block ID (* , 2) to (* , 5) - Used by Aspen Plus (* , 6) - User pressure drop subroutine name (* , 7) - User heat transfer subroutine name (* , 8) - User holdup subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector
XLONG	I	REAL*8	—	Reactor Length, specified on the RPlug Setup Configuration sheet (m)
DIAM	I	REAL*8	—	Reactor (or tube) diameter, specified on the RPlug Setup Specification sheet (m)

Variable	I/O/W [†]	Type	Dimension	Description
VOID	I	REAL*8	—	Void fraction from the RPlug Setup Catalyst sheet
THETA	I	REAL*8	—	Angle of pipe/tube from the horizontal (rad)
Z	I	REAL*8	—	Current reactor location (m)
LAMBDA	I	REAL*8	—	No slip liquid volume fraction (-)
SIGMA	I	REAL*8	—	Liquid surface tension (N/m)
VISL	I	REAL*8	—	Liquid viscosity (N-s/m ²)
VISG	I	REAL*8	—	Gas viscosity (N-s/m ²)
DENL	I	REAL*8	—	Liquid density (kg/m ³)
DENG	I	REAL*8	—	Gas density (kg/m ³)
VELMIX	I	REAL*8	—	Mixture velocity (m/s)
KFLASH	I	INTEGER	—	Stream flash flag: KFLASH = 0, stream has not been flashed KFLASH = 1, stream has been flashed Always set to 1 for RPlug
HOLDUP	O	REAL*8	—	Calculated volumetric liquid holdup fraction, liquid volume/total volume

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed for the process stream; NBOPST(1, 2) should be passed for the coolant stream.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **RPlug | User Subroutine | Pressure Drop** sheet (pressure drop subroutine) or the **RPlug | User Subroutine | Holdup** sheet (holdup subroutine). You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Pressure

For Calculation Option on the **RPlug | User Subroutine | Pressure Drop** sheet, select:

- **Calculate Pressure** if the user subroutine calculates pressure.
- **Calculate Pressure Drop** if the user subroutine calculates the rate of change of pressure with respect to reactor length.

Your subroutine should then return either the pressure or the differential pressure drop (dP/dz) for PRESF and PRESC.

Local Work Arrays

You can use local work arrays by specifying the array length on the **RPlug | User Subroutine | Pressure Drop** sheet (pressure drop subroutine) or the **RPlug | User Subroutine | Holdup** sheet (holdup subroutine). Aspen Plus does not retain these arrays from one call to the next.

User Variables (Pressure Drop Subroutine only)

User variables provide a convenient method to access intermediate variables calculated within user subroutines. For example, a user pressure drop subroutine may calculate the process fluid Reynolds number in order to calculate the pressure drop. This intermediate variable can be stored in the **USERV** array by the pressure drop subroutine.

User variables are identified with user-specified labels. These labels are specified on the **User Subroutine | User Variables** sheet. The **Units label** is used for convenience; Aspen Plus does not perform unit conversion on the user variables.

User variable profiles are tabulated in the reactor model's **Profiles | User Variables** sheet.

User variables can also be tabulated in the simulation history file by setting the **Simulation level** higher than 5 on the **Block Options | Diagnostics** sheet.

COMMON RPLG_RPLUGI

COMMON block RPLG_RPLUGI contains integer configuration parameters for the RPlug block. To use RPLG_RPLUGI, include the following directive (in column 1) of your subroutine:

```
#include "rplg_rplugi.cmn"
```

The following table describes the variables in RPLG_RPLUGI.

Variable	Type	Description
RPLUGI_IRPLUG	INTEGER	RPlug reactor type
RPLUGI_NTUBE	INTEGER	Number of reactor tubes

Reactor types:

- 1 = Reactor with Specified Temperature
- 2 = Adiabatic Reactor
- 3 = Reactor with Co-current Coolant
- 4 = Reactor with Counter-current Coolant
- 5 = Adiabatic Reactor (with substreams at different temperatures)
- 6 = Reactor with Co-current Coolant (with substreams at different temperatures)
- 7 = Reactor with Counter-current Coolant (with substreams at different temperatures)

- 8 = Reactor with Constant Coolant Temperature
- 9 = Reactor with Constant Coolant Temperature (with substreams at different temperatures)

COMMON RPLG_RPLUGR

COMMON block RPLG_RPLUGR contains real configuration parameters for the RPlug block. To use RPLG_RPLUGR, include the following directive (in column 1) of your subroutine:

```
#include "rplg_rplugr.cmn"
```

The following table describes the variables in RPLG_RPLUGR.

Variable	Type	Description
RPLUGR_UXLONG	REAL*8	Reactor length (m)
RPLUGR_UDIAM	REAL*8	Reactor (or tube) diameter (m)
RPLUGR_AXPOS	REAL*8	Current axial position (m)
RPLUGR_TCOOL	REAL*8	Coolant temperature (from Setup Specifications sheet) (K)
RPLUGR_CATWT	REAL*8	Total catalyst mass (kg)
RPLUGR_CAT_RHO	REAL*8	Catalyst mass density (kg/m ³)
RPLUGR_BED_VOID	REAL*8	Reactor bed void fraction (unitless)

13 User Heat Transfer Subroutine for RPlug

You can supply a heat transfer rate subroutine for the unit operation model RPlug. The user heat transfer subroutine calculates the heat transfer rates per unit reactor wall area at any point along the reactor.

You can specify the user heat transfer subroutine on the **RPlug | User Subroutine | Heat Transfer** sheet.

RPlug Heat Transfer Subroutine

Calling Sequence for RPlug Heat Transfer Subroutine

SUBROUTINE subrname[†] (SIN, SOUT, SINC, SOUTC, NSUBS, IDXSUB, ITYPE, NINTQ, INTQ, NREALQ, REALQ, IDS, NPO, NBOPST, NIWQ, IWORKQ, NWQ, WORKQ, QTCP, QTCA, QTCS, QTPA, Z, XLONG, DIAM, ANGLE, ELEV, ROUGH, AFRAC, NTUBE, VOID, DPART, SPHER, NUSERV, USERV)

[†] Subroutine name entered on the **RPlug | User Subroutine | Heat Transfer** sheet.

Argument List for RPlug Heat Transfer Subroutine

Variable	I/O [†]	Type	Dimension	Description
SIN	I	REAL*8	(1)	Inlet process stream vector (see Appendix C)
SOUT	I/O	REAL*8	(1)	Process stream vector at current reactor location Z (see Appendix C)
SINC	I	REAL*8	(1)	Inlet coolant stream vector (see Appendix C)
SOUTC	I/O	REAL*8	(1)	Coolant stream vector at current reactor location Z (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINTQ	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INTQ	I/O	INTEGER	NINTQ	Vector of integer parameters (see Integer and Real Parameters)
NREALQ	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REALQ	I/O	REAL*8	NREALQ	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block IDs (* , 1) - Block ID (* , 2) to (* , 5) - Used by Aspen Plus (* , 6) - User pressure drop subroutine name (* , 7) - User heat transfer subroutine name (* , 8) - User holdup subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWQ	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORKQ	W	INTEGER	NIWQ	Integer work vector
NWQ	I	INTEGER	—	Length of real work vector (see Local Work Arrays)

WORKQ	W	REAL*8	NWQ	Real work vector
QTCP	O	REAL*8	—	Heat transfer flux from coolant stream to process stream (W/m ²)
QTCA	O	REAL*8	—	(1) Heat transfer flux from coolant stream to ambient environment (W/m ²) ^{††} (2) Heat transfer flux from coolant stream to mixed substream of process stream (W/m ²) ^{††}
(QTCM) ^{††}				
QTCS ^{††}	O	REAL*8	—	(1) Not used ^{††} (2) Heat transfer flux from coolant stream to solid substreams of process stream (W/m ²) ^{††}
QTPA	O	REAL*8	—	(1) Heat transfer flux from the process stream to the ambient environment (W/m ²) ^{††} (2) Heat transfer flux from the solid substream to the mixed substream of the process stream (W/m ²) ^{††}
(QTSM) ^{††}				
Z	I	REAL*8	—	Current reactor location (m)
XLONG	I	REAL*8	—	Length of reactor you specified on the RPlug Setup Configuration sheet (m)
DIAM	I	REAL*8	—	Diameter of reactor you specified on the RPlug Setup Configuration sheet (m)
ANGLE	I	REAL*8	—	Angle of inclination (radians) from the RPlug Setup Configuration sheet
ELEV	I	REAL*8	—	Elevation change from feed to discharge (m) from the RPlug Setup Configuration sheet
ROUGH	I	REAL*8	—	Roughness (m) from the RPlug Setup Pressure sheet
AFRAC	I	REAL*8	—	Fraction of cross-sectional area occupied by the condensed phases
NTUBE	I	REAL*8	—	Number of tubes from the RPlug Setup Configuration sheet
VOID	I	REAL*8	—	Void fraction from the RPlug Setup Catalyst sheet
DPART	I	REAL*8	—	Particle diameter from the RPlug Setup Catalyst sheet (m)
SPHER	I	REAL*8	—	Particle sphericity from the RPlug Setup Catalyst sheet
NUSERV	I	INTEGER	—	Number of user variables from the RPlug User Subroutine User Variables sheet
USERV	I/O	REAL*8	NUSERV	Vector of user variables from the RPlug User Subroutine User Variables sheet

† I = Input to subroutine, O = Output from subroutine, W = Workspace

†† Option (1) is always used when the process stream does not include solid substreams.

Option (2) is only used when the process stream includes solid substreams and the RPlug block is configured to allow the solid and mixed substreams to be at different temperatures. See *Heat Flux Terms*, page 191.

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed for the process stream; NBOPST(1, 2) should be passed for the coolant stream.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **RPlug | User Subroutine | Heat Transfer** sheet. You can initialize these parameters by assigning values on the same sheet. Default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **RPlug | User Subroutine | Pressure Drop** sheet. Aspen Plus does not retain these arrays from one call to the next.

User Variables

User variables provide a convenient method to access intermediate variables calculated within user subroutines. For example, a user heat transfer subroutine may calculate values for the Reynolds and Prandtl numbers in order to calculate the overall heat transfer rates in the reactor.

User variables are identified with user-specified labels. These labels are specified on the **User Subroutine | User Variables** sheet. The **Units label** is used for convenience; Aspen Plus does not perform unit conversion on the user variables.

User variable profiles are tabulated in the reactor model's **Profiles | User Variables** sheet.

User variables can also be tabulated in the simulation history file by setting the **Simulation level** higher than five on the **Block Options | Diagnostics** sheet.

COMMON RPLG_RPLUGI

COMMON block RPLG_RPLUGI contains integer configuration parameters for the RPlug block. To use RPLG_RPLUGI, include the following directive (in column 1) of your subroutine:

```
#include "rplg_rplugi.cmn"
```


The following table describes the variables in RPLG_RPLUGI.

Variable	Type	Description
RPLUGI_IRPLUG	INTEGER	RPlug reactor type
RPLUGI_NTUBE	INTEGER	Number of reactor tubes

Reactor types:

- 1 = Reactor with Specified Temperature
- 2 = Adiabatic Reactor
- 3 = Reactor with Co-current Coolant
- 4 = Reactor with Counter-current Coolant
- 5 = Adiabatic Reactor (with substreams at different temperatures)
- 6 = Reactor with Co-current Coolant (with substreams at different temperatures)
- 7 = Reactor with Counter-current Coolant (with substreams at different temperatures)
- 8 = Reactor with Constant Coolant Temperature
- 9 = Reactor with Constant Coolant Temperature (with substreams at different temperatures)

COMMON RPLG_RPLUGR

COMMON block RPLG_RPLUGR contains real configuration parameters for the RPlug block. To use RPLG_RPLUGR, include the following directive (in column 1) of your subroutine:

```
#include "rplg_rplugr.cmn"
```

The following table describes the variables in RPLG_RPLUGR.

Variable	Type	Description
RPLUGR_UXLONG	REAL*8	Reactor length (m)
RPLUGR_UDIAM	REAL*8	Reactor (or tube) diameter (m)
RPLUGR_AXPOS	REAL*8	Current axial position (m)
RPLUGR_TCOOL	REAL*8	Coolant temperature (from Setup Specifications sheet) (K)
RPLUGR_CATWT	REAL*8	Total catalyst mass (kg)
RPLUGR_CAT_RHO	REAL*8	Catalyst mass density (kg/m ³)
RPLUGR_BED_VOID	REAL*8	Reactor bed void fraction (unitless)

Heat Flux Terms

The heat transfer model returns the net heat flux per unit wall area. The wall area can be calculated from the number of tubes (RPLUGI_NTUBE), the tube length (RPLUGR_UXLONG), and tube diameter (RPLUGR_UDIAM). The RPLUG model does not consider tube thickness.

In most cases, the heat transfer subroutine should calculate three heat flux terms:

- QTCP** Heat flux from the coolant stream to the process stream
- QTCA** Heat flux from the coolant stream to the environment (heat loss)
- QTPA** Heat flux from the process stream to the environment (heat loss)

The heat flux terms are all set to zero before the heat transfer routine is called. If these values need to be retained between calls they should be saved in the WORKQ vector.

When multiple substreams are present in RPlug, the Setup Specifications sheet displays the option to allow the solid and mixed substreams to be at different temperatures. When this option is chosen, the option number stored in RPLUGI_IRPLUG is 5, 6, 7, or 9, and the definitions of some of the arguments above are changed:

- QTCP** Heat flux from the coolant stream to the process stream
- QTCA (QTCM)** Heat flux from the coolant stream to the mixed substream
- QTCS** Heat flux from the coolant stream to the solid substream of the process stream
- QTPA (QTSM)** Heat flux from the solid substream of the process stream to the mixed substream

14 User Heat Transfer Subroutine for RBatch

You can supply a user subroutine for the unit operation model RBatch to calculate heat transfer rates. The user heat transfer subroutine calculates the instantaneous heat duty transferred to or from the reactor contents by convective and radiant heat transfer via a cooling jacket or a serpentine.

To supply a user heat transfer subroutine for RBatch, select **Heat Transfer User Subroutine** for the **Reactor Operations** specification on the **RBatch | Setup | Specifications** sheet, and then specify the subroutine name on the **RBatch | User Subroutine | Heat Transfer** sheet.

RBatch Heat Transfer Subroutine

Calling Sequence for RBatch Heat Transfer Subroutine

SUBROUTINE subrname[†] (SOUT, NSUBS, IDXSUB, ITYPE, NINT, INT, NREAL, REAL, IDS, NPO, NBOPST, NIWORK, IWORK, NWORK, WORK, TIME, DUTY, NUSERV, USERV)

[†] Subroutine name you entered on the **RBatch | User Subroutine | Heat Transfer** sheet.

Argument List for RBatch Heat Transfer Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SOUT	I/O	REAL*8	(1)	Stream vector for reactor contents at current time (see Stream Structure)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector: 1=MIXED, 2=CISOLID, 3=NC
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block Ids (* , 1) - Block ID (* , 2) to (* , 5) - Used by Aspen Plus (* , 6) - Heat transfer subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 1)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector
TIME	I	REAL*8	—	Current time (sec)
DUTY	O	REAL*8	—	Heat transfer rate at current time (positive for heating, negative for cooling) (watt)

Variable	I/O/W [†]	Type	Dimension	Description
NUSERV	I	INTEGER	-	Number of user variables from the RBatch User Subroutine User Variables sheet
USERV	I/O	REAL*8	NUSERV	Vector of user variables from the RBatch User Subroutine User Variables sheet

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

Stream Structure

The stream structure for material streams is described in Appendix C. The stream vector for the reactor contents represents the actual molar or mass amounts of the components in the reactor (all phases), not their flow rates.

NBOPST

NBOPST is used when calling FLSH_FLASH or a property monitor.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **RBatch | User Subroutine | Heat Transfer** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **RBatch | User Subroutine | Heat Transfer** sheet. Aspen Plus does not retain these arrays from one call to the next.

User Variables

User variables provide a convenient method to access intermediate variables calculated within user subroutines. For example, a user heat transfer subroutine may calculate values for the Reynolds and Prandtl numbers in order to calculate the overall heat transfer rates in the reactor.

User variables are identified with user-specified labels. These labels are specified on the **User Subroutine | User Variables** sheet. The **Units label** is used for convenience; Aspen Plus does not perform unit conversion on the user variables.

User variable profiles are tabulated in the **RBatch | Profiles | User Variables** sheet.

User variables can also be tabulated in the simulation history file by setting the **Simulation level** higher than 5 on the **Block Options | Diagnostics** sheet.

15 User Subroutines for RYield

You can supply user subroutines for the unit operation model RYield. A yield subroutine calculates the yields and component flows for each component in each substream, while a petroleum characterization subroutine re-characterizes the pre-defined assay or blend fed to the block. In either case, the results of the subroutine are used to fill the outlet stream of the RYield block.

The yield subroutine can also change the particle size distribution (PSD) and component attributes for the outlet stream. For example, if coal is devolatilized, its analysis, as represented in Aspen Plus by component attributes such as ultimate and proximate analysis, must change.

Use the **RYield | User Subroutine | Yield** sheet to specify a user yield subroutine. Use the **RYield | Assay Analysis | Subroutine** sheet to specify a user petroleum characterization subroutine.

RYield User Subroutines

Calling Sequence for RYield Yield Subroutine

SUBROUTINE subrname[†] (SIN, SOUT, NSUBS, IDXSUB, ITYPE, NINT, INT, NREAL, REAL, IDS, NPO, NBOPST, NIWORK, IWORK, NWORK, WORK)

[†] Subroutine name you entered on the **RYield | User Subroutine | Yield** sheet.

Argument List Descriptions for RYield Yield Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SIN	I	REAL*8	(1)	Inlet stream (see Appendix C)
SOUT	O	REAL*8	(1)	Outlet stream (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block Ids (* , 1) - Block ID (* , 2) to (* , 4) - Used by Aspen Plus (* , 5) - Yield subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 1)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector (see Local Work Arrays)

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

Calling Sequence for RYield Petroleum Characterization Subroutine

SUBROUTINE subrname[†] (SID, NTBPPT, TBPPCT, TBP, NGRPT, GRBULK, GRTEMP, GRVAL, NMWPT, MWBULK, MWTEMP, MWVAL, NPCURV, NPPT, PRNAME, PRTEMP, PRVAL, NVCURV, NVPT, VISBT, VISTEM, VISVAL, NTBMAX, NGRMAX, NMWMAX, NPMAX, NPCMAX, NVCMAX, NIWORK, IWORK, NWORK, WORK)

[†] Subroutine name you entered on the **RYield | Assay Analysis | Subroutine** sheet.

Argument List Descriptions for RYield Petroleum Characterization Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SID	I	INTEGER	—	Outlet stream name for petroleum characterization
NTBPPT	I/O	INTEGER	—	Number of data points in TBP curve
TBPPCT	O	REAL*8	NTBMAX	Percent distilled in TBP curve
TBP	O	REAL*8	NTBMAX	Temperature in TBP curve
NGRPT	I/O	INTEGER	—	Number of data points in gravity curve
GRBULK	O	REAL*8	—	Bulk gravity value
GRTEMP	O	REAL*8	NGRMAX	Distillation temperature in gravity curve
GRVAL	O	REAL*8	NGRMAX	Gravity value in gravity curve
NMWPT	I/O	INTEGER	—	Number of data points in gravity curve
MWBULK	O	REAL*8	—	Bulk molecular weight value
MWTEMP	O	REAL*8	NMWMAX	Distillation temperature in molecular weight curve
MWVAL	O	REAL*8	NMWMAX	Molecular weight value in molecular weight curve
NPCURV	I/O	INTEGER	—	Number of petroleum property curves, excluding viscosity curves
NPPT	O	INTEGER	NPCMAX	Number of data points of petroleum property curves
PRNAME	O	INTEGER	2, NPCMAX	Petroleum property names
PRTEMP	O	REAL*8	NPMAX, NPCMAX	Distillation temperature in petroleum property curves
PRVAL	O	REAL*8	NPMAX, NPCMAX	Property value in petroleum property curves
NVCURV	I/O	INTEGER	—	Number of viscosity curves
NVPT	O	INTEGER	NVCMAX	Number of data points of viscosity curves
VISBT	O	REAL*8	NVCMAX	Reference temperature
VISTEM	O	REAL*8	NPMAX, NVCMAX	Distillation temperature in viscosity curves
VISVAL	O	REAL*8	NPMAX, NVCMAX	Viscosity value in viscosity curves

NTBMAX	I	INTEGER	—	Maximum number of data points allowed for TBP curve
NGRMAX	I	INTEGER	—	Maximum number of data points allowed for gravity curve
NMWMAX	I	INTEGER	—	Maximum number of data points allowed for molecular weight curve
NPMAX	I	INTEGER	—	Maximum number of data points allowed for property curves, including viscosity curves
NPCMAX	I	INTEGER	—	Maximum number of property curves allowed for a stream, excluding viscosity curves
NVCMAX	I	INTEGER	—	Maximum number of viscosity curves allowed for a stream
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NWORK	Real work vector (see Local Work Arrays)

† I = Input to subroutine, O = Output from subroutine, W = Workspace

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed to the yield subroutine.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **RYield | User Subroutine | Yield** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **RYield | User Subroutine | Yield** sheet. Aspen Plus does not retain these arrays from one call to the next.

16 User KLL Subroutines

You can supply a user subroutine to calculate liquid-liquid equilibrium K-values (distribution coefficients) for the unit operation models Extract, RadFrac, and Decanter. User KLL subroutines are useful when you have an empirical correlation for the distribution coefficients. This table lists the sheets where you specify the KLL subroutine name:

Unit operation model Sheet

Decanter	Decanter Properties KLL Subroutine
Extract	Extract Properties KLL Subroutine
RadFrac	RadFrac User Subroutines KLL

The liquid-liquid equilibrium K-value of component i , K_i , is defined such that at equilibrium:

$$K_i(T, P, \underline{x}^I, \underline{x}^{II}) = \frac{x_i^{II}}{x_i^I}$$

Where:

T = Temperature

P = Pressure

\underline{x}^I = Mole fraction vector for liquid phase 1

\underline{x}^{II} = Mole fraction vector for liquid phase 2

x_i^{II} = Mole fraction of component i in liquid phase 2

x_i^I = Mole fraction of component i in liquid phase 1

The user subroutine calculates K_i for each component given values for T , P , \underline{x}^I and \underline{x}^{II} .

User KLL Subroutine

Calling Sequence for User KLL Subroutines

SUBROUTINE subrname[†] (T, P, X1, X2, N, IDX, NBOPST, KDIAG, KBASE, RK, RDUM, KER, NINT, INT, NREAL, REAL, Istage, IBasis)

[†] Subroutine name you entered on a sheet listed on page 201.

Argument List Descriptions for User KLL Subroutines

Variable	I/O [†]	Type	Dimension	Description
T	I	REAL*8	—	Temperature (K)
P	I	REAL*8	—	Pressure (N/m ²)
X1	I	REAL*8	N	Mole fraction vector for liquid phase 1 (see X1, X2)
X2	I	REAL*8	N	Mole fraction vector for liquid phase 2 (see X1, X2)
N	I	INTEGER	—	Number of components present (see IDX)
IDX	I	INTEGER	N	Component index vector for X1, X2, and RK (see IDX)
NBOPST	I	INTEGER	6	Physical property option set vector (see NBOPST, KDIAG)
KDIAG	I	INTEGER	—	Property diagnostic level (see NBOPST, KDIAG)
KBASE	I	INTEGER	—	Thermodynamic function base code: 0, pure components in ideal gas state at 298.15K; 1, elements in their standard states at 298.15K
RK	O	REAL*8	N	Vector of Liquid-Liquid K-values (see X1, X2)
RDUM	—	—	—	Not used
KER	O	INTEGER	—	Error return code: ≠0 if an error or warning condition occurred =0 otherwise
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
Istage	I	INTEGER	—	Stage Number (only for RadFrac, Extract)
IBasis	O	INTEGER	—	Basis for the calculated K-values: 0 = Mole; 1 = Mass; 2 = Standard Volume

[†] I = Input to subroutine, O = Output from subroutine

IDX

IDX is a vector containing component sequence numbers of the conventional components actually present in the order that they appear on the Components Specification Selection sheet. For example, if only the first and third conventional components are present, $N=2$ and $IDX=(1,3)$. RK should be filled in only for the components actually present.

X1, X2

X1 and X2 are packed mole fraction vectors of length N containing the mole fractions of the components that are present and are to be included in the calculations. The order of the components is defined by the IDX vectors. For example, if $N=2$ and $IDX=(1,3)$, $X1(2)$ is the liquid 1 mole fraction of the conventional component listed third on the Components Specifications Selection sheet.

NBOPST, KDIAG

Your subroutine should use the variables NBOPST and KDIAG only if Aspen Plus physical property monitors are called in the user subroutine. These variables should be passed to the Aspen Plus subroutine. For more information about using monitors, see Chapter 3.

Component Sequence Number

It is sometimes desirable to make a user KLL subroutine independent of the order in which components are entered on the Components Specifications Selection sheet. Use the utility DMS_KFORM or DMS_KFORMC (see Chapter 4) to find the sequence number of a component in the set of conventional components you entered on the Components Specifications Selection sheet.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on a sheet listed on page 16-1. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

17 User Subroutines for Pipes and HeatX

This chapter describes the user pressure drop and liquid holdup subroutines for Pipeline, Pipe, and HeatX, and the user diameter subroutine for Pipe.

To supply a user pressure drop and liquid holdup subroutine for Pipeline or Pipe:

- Select **User Subroutine** for **Downhill, Inclined, Horizontal,** and **Vertical** on the **Pipeline | Setup | Methods** sheet or **Pipe | Advanced | Methods** sheet.
- Specify the subroutine names on the **Pipeline** or **Pipe | User Subroutine | Pressure Drop** and **Liquid Holdup** sheets.

To supply a user pressure drop and liquid holdup subroutine for the tube side of a shell-and-tube HeatX:

- Select **User Subroutine** for **Pressure Drop Correlation** or **Liquid Holdup Correlation** on the **HeatX | Setup | Pressure Drop** sheet.
- Specify the subroutine names on the **HeatX | User Subroutine | Pressure Drop** sheet or the **HeatX | User Subroutine | Liquid Holdup** sheet.

You can supply either or both of the pressure drop and liquid holdup subroutines.

If you supply both subroutines, Aspen Plus will call the liquid holdup subroutine first.

To specify a user diameter subroutine for Pipe:

- Select **Compute using user subroutine** for **Diameter** on the **Pipe | Setup | Pipe Parameters** sheet.
- Specify the subroutine name on the **Pipe | User Subroutine | Diameter** sheet.

User Pressure Drop Subroutine

Calling Sequence for User Pressure Drop Subroutine

SUBROUTINE subrname[†] (SOUT, NSUBS, IDXSUB, ITYPE, NINT, INT, NREAL, REAL, IDS, NPO, NBOPST, NIWORK, IWORK, NWORK, WORK, XLEN, DIAM, THETA, RELRUF, Z, XLFR, XL2FR, SIGMA, VISL, VISG, DENL, DENG, VELMIX, VELSON, VELSG, VELSL, KFLASH, IREGIM, HOLDUP, DPFRI, DPAC, DPTOT)

[†] Subroutine name you entered on the **User Subroutine | Pressure Drop** sheet.

Argument List Descriptions, User Pressure Drop Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SOUT	I	REAL*8	(1)	Stream vector (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block ID and subroutine names: (* ,1) - Block ID (* ,2) to (* ,4) - Used by Aspen Plus (* ,5) - User holdup subroutine name (* ,6) - User pressure drop subroutine name (* ,7) - User heat transfer coefficient subroutine name (HeatX) (* ,8) - User LMTD correction subroutine name (HeatX)
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NIWORK	Real work vector (see Local Work Arrays)

Variable	I/O/W [†]	Type	Dimension	Description	
XLEN	I	REAL*8	—	Pipe/tube length (m)	
DIAM	I	REAL*8	—	Pipe/tube diameter (m)	
THETA	I	REAL*8	—	Angle of pipe/tube from the horizontal (rad)	
RELRUF	I	REAL*8	—	Pipe/tube relative roughness (absolute roughness/DIAM) (dimensionless)	
Z	I	REAL*8	—	Current pipe/tube location (m)	
XLFR	I	REAL*8	—	Volumetric ratio of the liquid phase to total (all phases combined)	
XL2FR	I	REAL*8	—	Volumetric ratio of the second liquid phase to the combined liquid phase	
SIGMA	I	REAL*8	—	Liquid surface tension (N/m)	
VISL	I	REAL*8	—	Liquid viscosity (N-s/m ²)	
VISG	I	REAL*8	—	Gas viscosity (N-s/m ²)	
DENL	I	REAL*8	—	Liquid density (kg/m ³)	
DENG	I	REAL*8	—	Gas density (kg/m ³)	
VELMIX	I	REAL*8	—	Mixture velocity (m/s)	
VELSON	I	REAL*8		Velocity of sound (m/s)	
VELSG	I	REAL*8	—	Superficial gas velocity (m/s)	
VELSL	I	REAL*8	—	Superficial liquid velocity (m/s)	
KFLASH	I	INTEGER	—	Stream flash flag: KFLASH = 0, stream has not been flashed (possible for Pipeline and Pipe only) KFLASH = 1, stream has been flashed (see KFLASH)	
IREGIM	I	INTEGER	—	Flow regime, taken from the list below:	
1-Segregated		5-Mist		9-Annular	13-Heading
2-Distributed		6-Bubble		10-Liquid	14-Wave
3-Stratified		7-Intermittent		11-Vapor	15-Disp. Bubb
4-Slug		8-Transition		12-Undefined	
HOLDUP	I	REAL*8	—	Calculated volumetric liquid holdup fraction, as: liquid volume/total volume	
DPFRIC	O	REAL*8	—	Length derivative of frictional pressure drop (N/m ² /m)	
DPEL	I/O	REAL*8	—	Length derivative of elevational pressure drop (N/m ² /m)	
DPACC	I/O	REAL*8	—	Length derivative of accelerational pressure drop (N/m ² /m)	
DPTOT	O	REAL*8	—	Length derivative of total pressure drop (N/m ² /m). Initialized to USER_RUMISS, located in COMMON/PPEXEC_USER/ (see Appendix A). If not set by the user subroutine, DPTOT is calculated as the sum of DPFRIC, DPEL, and DPACC.	

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

User Liquid Holdup Subroutine

Calling Sequence for User Liquid Holdup Subroutine

```
SUBROUTINE subrname†      (SOUT, NSUBS, IDXSUB, ITYPE, NINT, INT,
                             NREAL, REAL, IDS, NPO, NBOPST, NIWORK,
                             IWORK, NWORK, WORK, XLEN, DIAM, THETA,
                             RELRUF, Z, XLFR, XL2FR, SIGMA, VISL,
                             VISG, DENL, DENG, VELMIX, VELSON, VELSG,
                             VELSL, KFLASH, IREGIM, HOLDUP)
```

[†] Subroutine name you entered on the **User Subroutine | Liquid Holdup** sheet.

Argument List Descriptions, User Liquid Holdup Subroutine

Variable	I/O [†]	Type	Dimension	Description
SOUT	I	REAL*8	(1)	Stream vector
NSUBS	I	INTEGER	—	Number of substreams
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Block Ids: (I, 1) - Block ID (I, 2) to (I, 4) - Used by Aspen Plus (I, 5) - User holdup subroutine name (I, 6) - User pressure drop subroutine name
NPO	I	INTEGER	—	Number of property option sets (always 2)
NBOPST	I	INTEGER	6, NPO	Property option set array
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NIWORK	Real work vector (see Local Work Arrays)
XLEN	I	REAL*8	—	Pipe/tube length (m)
DIAM	I	REAL*8	—	Pipe/tube diameter (m)
THETA	I	REAL*8	—	Angle of pipe/tube from the horizontal (rad)

Variable	I/O [†]	Type	Dimension	Description	
RELRF	I	REAL*8	—	Pipe/tube relative roughness. Absolute roughness/DIAM. Dimensionless.	
Z	I	REAL*8	—	Current pipe/tube location (m)	
XLFR	I	REAL*8	—	Volumetric ratio of the liquid phase to total (all phases combined)	
XL2FR	I	REAL*8	—	Volumetric ratio of the second liquid phase to the combined liquid phase	
SIGMA	I	REAL*8	—	Liquid surface tension (N/m)	
VISL	I	REAL*8	—	Liquid viscosity (N-s/m ²)	
VISG	I	REAL*8	—	Gas viscosity (N-s/m ²)	
DENL	I	REAL*8	—	Liquid density (kg/m ³)	
DENG	I	REAL*8	—	Gas density (kg/m ³)	
VELMIX	I	REAL*8	—	Mixture velocity (m/s)	
VELSON	I	REAL*8	—	Velocity of sound (m/s)	
VELSG	I	REAL*8	—	Superficial gas velocity (m/s)	
VELSL	I	REAL*8	—	Superficial liquid velocity (m/s)	
KFLASH	I	INTEGER	—	Stream flash flag: KFLASH = 0, stream has not been flashed (possible for Pipeline and Pipe only) KFLASH = 1, stream has been flashed (see KFLASH)	
IREGIM	O	INTEGER	—	Flow regime, taken from the list below:	
1-Segregated		5-Mist		9-Annular	13-Heading
2-Distributed		6-Bubble		10-Liquid	14-Wave
3-Stratified		7-Intermittent		11-Vapor	15-Disp. Bubb
4-Slug		8-Transition		12-Undefined	
HOLDUP	O	REAL*8	—	Calculated volumetric liquid holdup fraction, liquid volume/total volume	

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

User Diameter Subroutine

Calling Sequence for User Diameter Subroutine

```
SUBROUTINE subrname† (SOUT, NSUBS, IDXSUB, ITYPE, NINT,
                        INT, NREAL, REAL, IDS, NPO,
                        NBOPST, NIWORK, IWORK, NWORK, WORK,
                        DIAM, DIAMN, ABSRUF, XLFR, XL2FR,
                        SIGMA, VISL, VISG, DENL, DENG,
                        KFLASH, IERR)
```

[†] Subroutine name you entered on the **User Subroutine | Diameter** sheet.

Argument List Descriptions, User Diameter Subroutine

Variable	I/O/W [†]	Type	Dimension	Description
SOUT	I	REAL*8	(1)	Stream vector (see Appendix C)

Variable	I/O/W [†]	Type	Dimension	Description
NSUBS	I	INTEGER	—	Number of substreams in stream vector
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector (1-MIXED, 2-CISOLID, 3-NC)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
IDS	I	INTEGER	2, 13	Blocks IDs: (* ,1) - Block ID (* ,2) to (* ,4) - Used by Aspen Plus (* ,5) - User holdup subroutine name (* ,6) - User pressure drop subroutine name
NPO	I	INTEGER	—	Number of property option sets
NBOPST	I	INTEGER	6, NPO	Property option set array (see NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	W	INTEGER	NIWORK	Integer work vector (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	W	REAL*8	NIWORK	Real work vector (see Local Work Arrays)
DIAM	O	REAL*8	—	Pipe inner diameter (m)
DIAMN	O	REAL*8	—	Nominal pipe diameter (m)
ABSRUF	I	REAL*8	—	Pipe absolute roughness (m)
SIGMA	I	REAL*8	—	Liquid surface tension (N/m)
XLFR	I	REAL*8	—	Volumetric ratio of the liquid phase to total (all phases combined)
XL2FR	I	REAL*8	—	Volumetric ratio of the second liquid phase to the combined liquid phase
VISL	I	REAL*8	—	Liquid viscosity (N-s/m ²)
VISG	I	REAL*8	—	Gas viscosity (N-s/m ²)
DENL	I	REAL*8	—	Liquid density (kg/m ³)
DENG	I	REAL*8	—	Gas density (kg/m ³)
KFLASH	I	INTEGER	—	Stream flash flag: KFLASH = 0, stream has not been flashed KFLASH = 1, stream has been flashed (see KFLASH)
IERR	O	INTEGER	—	Error flag (printed in report)

[†] I = Input to subroutine, O = Output from subroutine, W = Workspace

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the sheet where you specify the subroutine name. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

NBOPST

When calling FLSH_FLASH or a property monitor, NBOPST should be passed.

Local Work Arrays

You can use local work arrays by specifying the array length on the sheet where you specify the subroutine name. Aspen Plus does not retain these arrays from one call to the next.

KFLASH

Pipeline and Pipe can either use a property table or perform stream flashes while integrating down the length of the pipe (specify **Interpolate from Property Grid** on the **Pipe | Advanced | Calculation Options** sheet or the **Pipeline | Setup | Configuration** sheet). If **Do Flash at Each Step** is selected, then KFLASH will be set to 1, and SOUT will be updated at the current pipe location. If **Interpolate from Property Grid** is selected, then KFLASH will be set to zero and the temperature, pressure, enthalpy, vapor and liquid fractions, and density values will be updated by interpolating within the property table. The entropy values will not be updated.

KFLASH will always be set to 1 for HeatX.

18 User Heat Transfer Subroutine for HeatX

You can supply a user subroutine for the unit operation model HeatX to calculate heat transfer coefficients.

To supply a user heat transfer subroutine for HeatX, select **User Subroutine** for **Calculation Method** on the **HeatX | Setup | U Methods** sheet and specify the subroutine name on the **HeatX | User Subroutines | Heat Transfer** sheet.

HeatX Heat Transfer Subroutine

Calling Sequence for HeatX Heat Transfer Subroutine

```
SUBROUTINE subrname† (SIN, SOUT, TIN, TOUT, NSUBS, IDXSUB,
                        TYPE, NINT, INT, NREAL, REAL, NIDS, IDS,
                        NPO, NBOPST, NIWORK, IWORK, NWORK, WORK,
                        IEQSP, REQSP, ITUBE, RTUBE, ISEGB, RSEGB,
                        IRODB, RRODB, RFINS, ISSTR, ICALC, UCALC)
```

[†]Subroutine name you entered on the **HeatX | User Subroutines | Heat Transfer** sheet.

Argument List Descriptions for HeatX Heat Transfer

Variable	I/O [†]	Type	Dimension	Description
SIN	I	REAL*8	(1)	Inlet shell stream
SOUT	I	REAL*8	(1)	Outlet shell stream
TIN	I	REAL*8	(1)	Inlet tube stream
TOUT	I	REAL*8	(1)	Outlet tube stream
NSUBS	I	INTEGER	—	Number of substreams
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector: 1=MIXED, 2=CISOLID, 3=NC
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)

Variable	I/O[†]	Type	Dimension	Description
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
NIDS	I	INTEGER	—	Dimension for IDS
IDS	I	INTEGER	2, NIDS	Block ID and subroutine names: (* ,1) - Block ID (* ,2) to (* ,4) - Used by Aspen Plus (* ,5) - User holdup subroutine name (* ,6) - User pressure drop subroutine name (* ,7) - User heat transfer coefficient subroutine name (* ,8) - User LMTD correction subroutine name
NPO	I	INTEGER	—	Number of property option sets (=2)
NBOPST	I	INTEGER	6, 2	Property option sets (* ,1) - Shell stream (* ,2) - Tube stream (See NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORK	I	INTEGER	NIWORK	Integer work vector
NWORK	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORK	I	REAL*8	NWORK	Real work vector
IEQSP	I	INTEGER	9	Integer array of equipment specs (see Equipment Specification Arrays)
REQSP	I	REAL*8	2	Real array of equipment spec data (see Equipment Specification Arrays)
ITUBE	I	INTEGER	5	Integer array of tube bank data (see Equipment Specification Arrays)
RTUBE	I	REAL*8	5	Real array of tube bank data (see Equipment Specification Arrays)
ISEGB	I	INTEGER	3	Integer array of segmental baffle shell data (see Equipment Specification Arrays)
RSEGB	I	REAL*8	17	Real array of segmental baffle shell data (see Equipment Specification Arrays)
IRODB	I	INTEGER	(1)	Integer array of rod baffle shell data (see Equipment Specification Arrays)
RRODB	I	REAL*8	11	Real array of rod baffle shell data (see Equipment Specification Arrays)
RFINS	I	REAL*8	7	Real array of low-fin data (see Equipment Specification Arrays)
ISSTR	I	INTEGER	—	Shell stream flag = 0 Shell stream is hot = 1 Shell stream is cold

Variable	I/O [†]	Type	Dimension	Description
ICALC	I	INTEGER	—	Heat transfer coefficient calculation flag (specified on User Subroutines HeatTransfer sheet) = 0 Overall coefficient = 1 Local coefficient
UCALC	O	REAL*8	—	Calculated heat transfer coefficient (W/m ² K)

[†] I = Input to subroutine, O = Output from subroutine

NBOPST

NBOPST is used when calling FLSH_FLASH or a property monitor.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the HeatX User Subroutines HeatTransfer sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the HeatX User Subroutines HeatTransfer sheet. Aspen Plus does not retain these arrays from one call to the next.

Equipment Specification Arrays

The following table explains the usage of the equipment specification arrays.

Equipment Specification Arrays Usage

Array	Entry	Form	Description
IEQSP	1	Setup Specifications	Flow direction = 0 Counter-current = 1 Co-current = 2 Crossflow
	2		(future use)
	3		(future use)
	4		(future use)
	5	Geometry Shell	Number of tube passes

Array	Entry	Form	Description
	6	Geometry Shell	TEMA shell type = 0 TEMA E shell = 1 TEMA F shell = 2 TEMA G shell = 3 TEMA H shell = 4 TEMA J shell = 5 TEMA X shell
	7	Geometry Baffles	Baffle type = 0 No baffles = 1 Segmental baffles = 2 Rod baffles
	8	Geometry Shell	Exchanger Orientation = 0 Horizontal = 1 Vertical
	9	Geometry Shell	Direction of tubeside flow = 0 Up flow = 1 Down flow
REQSP	1	Geometry Shell	Inside shell diameter (m)
	2	Geometry Shell	Shell to bundle clearance (m)
TUBE	1	Geometry Tubes	Total number of tubes
	2	(calculated)	Number of tube rows
	3	Geometry Tubes	Tube type = 0 Bare tubes = 1 Low-finned tubes
	4	Geometry Tubes	Tube layout pattern = 0 Triangle (30) = 1 Rotated square (45) = 1 Rotated triangle (60) = 2 Square (90)
	5	Geometry Tubes	Tube material
	6	Geometry Tubes	Nominal size (diameter) = 0 1/4" = 1 3/8" = 2 1/2" = 3 5/8" = 4 3/4" = 5 7/8" = 6 1" = 7 1-1/4" = 8 1-1/2" = 9 2" = 10 2-1/2"
	7	Geometry Tubes	Birmingham wire gauge
RTUBE	1	Geometry Tubes	Tube length (m)
	2	Geometry Tubes	Tube inner diameter (m)
	3	Geometry Tubes	Tube outer diameter (m)
	4	Geometry Tubes	Tube thickness (m)
	5	Geometry Tubes	Tube pitch (m)
	6	Geometry Tubes	Tube thermal conductivity (watts/m-K)
ISEGB	1	Geometry Baffles	Number of segmental baffles

Array	Entry	Form	Description
	2	Geometry Shell	Number of sealing strip pairs
	3	Geometry Shell	Tubes in window = 0 Yes = 1 No
RSEGB	1	Geometry Baffles	Baffle cut
	2	Geometry Baffles	Shell to baffle clearance (m)
	3	Geometry Baffles	Tube to baffle clearance (m)
	4	Geometry Baffles	Baffle spacing (m)
	5	Geometry Baffles	Tubesheet to first baffle spacing (m)
	6	Geometry Baffles	Last baffle to tubesheet spacing (m)
	7	(calculated)	Net crossflow area of one baffle window (m ²)
	8	(calculated)	Hydraulic diameter of baffle window (m)
	9	(calculated)	Fraction of tubes in crossflow between baffle tips
	10	(calculated)	Number of tube rows crossed in one baffle window
	11	(calculated)	Number of tube rows crossed between baffle tips
	12	(calculated)	Total number of tube rows crossed
	13	(calculated)	Crossflow area near shell centerline (m ²)
	14	(calculated)	Bypass area ratio
	15	(calculated)	Sealing strip ratio
	16	(calculated)	Total leakage area ratio
	17	(calculated)	Shell to baffle leakage ratio
IRODB	1	Geometry Baffles	Number of rod baffles
RRODB	1	Geometry Baffles	Rod baffle spacing (m)
	2	Geometry Baffles	Rod Baffle inside diameter of ring (m)
	3	Geometry Baffles	Rod Baffle outside diameter of ring (m)
	4	Geometry Baffles	Rod Baffle support rod diameter (m)
	5	Geometry Baffles	Rod Baffle total length of support rods per baffle (m)
	6	(calculated)	Parallel flow area (m ²)
	7	(calculated)	Leakage flow area (m ²)
	8	(calculated)	Leakage to shell parameter
	9	(calculated)	Baffle flow area (m ²)
	10	(calculated)	Hydraulic diameter (m)
	11	(calculated)	Parallel flow characteristic diameter (m)
FINS	1	Geometry Tube Fins	Ratio of finned area to inside tube area
	2	Geometry Tube Fins	Fin efficiency
	3	Geometry Tube Fins	Root diameter of finned tube (m)
	4	Geometry Tube Fins	Number of fins per unit length (1/m)
	5	Geometry Tube Fins	Fin thickness (m)
	6	Geometry Tube Fins	Fin height (m)
	7	(calculated)	Finned tube effective diameter (m)

19 User LMTD Correction Factor Subroutine for HeatX

You can supply a user subroutine for the unit operation model HeatX to calculate a log mean temperature difference correction factor.

To supply a user LMTD correction factor subroutine for HeatX, select **User-Subr** for **LMTD Correction Method** on the **HeatX | Setup | Specifications** sheet, and specify the subroutine name on the **HeatX | User Subroutines | LMTD** sheet.

HeatX LMTD Correction Factor Subroutine

Calling Sequence: HeatX LMTD Correction Factor

SUBROUTINE subrname[†] (SIN, SOUT, TIN, TOUT, NSUBS, IDXSUB, ITYPE, NINT, INT, NREAL, REAL, NIDS, IDS, NPO, NBOPST, NIWORK, IWORK, NWORK, WORK, IEQSP, ICFLW, UAVG, AREA, FMTD)

[†] Subroutine name you entered on the **HeatX | User Subroutines | LMTD** sheet.

Argument List for HeatX LMTD Correction Factor

Variable	I/O [†]	Type	Dimension	Description
SIN	I	REAL*8	(1)	Inlet shell stream
SOUT	I	REAL*8	(1)	Outlet shell stream
TIN	I	REAL*8	(1)	Inlet tube stream
TOUT	I	REAL*8	(1)	Outlet tube stream
NSUBS	I	INTEGER	—	Number of substreams
IDXSUB	I	INTEGER	NSUBS	Location of substreams in stream vector
ITYPE	I	INTEGER	NSUBS	Substream type vector: 1=MIXED, 2=CISOLID, 3=NC
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
NIDS	I	INTEGER	—	Dimension for IDS
IDS	I	INTEGER	2, NIDS	Blocks IDs: (* ,1) - Block ID (* ,2) to (* ,4) - Used by Aspen Plus (* ,5) - User holdup subroutine name (* ,6) - User pressure drop subroutine name (* ,7) - User heat transfer coefficient subroutine name (* ,8) - User LMTD correction subroutine name
NPO	I	INTEGER	—	Number of property option sets (=2)
NBOPST	I	INTEGER	6, 2	Property option sets (* ,1) - Shell stream (* ,2) - Tube stream (See NBOPST)
NIWORK	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)

IWORK	I	INTEGER NIWORK	Integer work vector
NWORK	I	INTEGER —	Length of real work vector (see Local Work Arrays)
WORK	I	REAL*8 NWORK	Real work vector
IEQSP	I	INTEGER 9	Integer array of equipment specs (see Equipment Specification Arrays)
ICFLW	I	INTEGER 6	Integer array of crossflow data (see Equipment Specification Arrays)
UAVG	I	REAL*8 —	Average heat transfer coefficient (W/m ² K)
AREA	I	REAL*8 —	Heat transfer area (m ²)
FMTD	O	REAL*8 —	LMTD correction factor

† I = Input to subroutine, O = Output from subroutine

NBOPST

NBOPST is used when calling FLSH_FLASH or a property monitor.

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **HeatX | User Subroutines | LMTD** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the HeatX User Subroutines LMTD sheet. Aspen Plus does not retain these arrays from one call to the next.

Equipment Specification Arrays

The following table explains the usage of the equipment specification arrays:

Array	Entry	Sheet	Description
IEQSP	1	Setup Specifications	Flow direction = 0 Counter-current = 1 Co-current = 2 Crossflow
	2		(Future use)
	3		(Future use)
	4		(Future use)
	5	Geometry Shell	Number of tube passes
	6	Geometry Shell	TEMA shell type = 0 TEMA E shell = 1 TEMA F shell = 2 TEMA G shell = 3 TEMA H shell = 4 TEMA J shell = 5 TEMA X shell
	7	Geometry Baffles	Baffle type = 0 No baffles = 1 Segmental baffles = 2 Rod baffles
	8	Geometry Shell	Exchanger Orientation = 0 Horizontal = 1 Vertical
	9	Geometry Shell	Direction of tubeside flow = 0 Up flow = 1 Down flow
ICFLW	1		(Future use)
	2		(Future use)
	3	Geometry Shell	Crossflow tubeside mixing = 0 Unmixed = 1 Between passes = 2 Throughout
	4		(Future use)
	5	Geometry Shell	Crossflow shellside mixing = 0 Unmixed = 1 Between passes = 2 Throughout
	6		(Future use)

20 User Subroutines for Rate-Based Distillation

The Aspen Rate-Based Distillation¹ feature in RadFrac can use Fortran subroutines you supply to calculate:

- Binary mass transfer coefficients for vapor and liquid phases
- Heat transfer coefficients for vapor and liquid phases
- Interfacial area
- Holdup for liquid and vapor phases

This chapter provides argument lists and values you need for user subroutines that perform all of these calculations. Use these sheets to specify the subroutines in Aspen Plus:

To calculate	Use sheet
Binary mass transfer coefficients	RadFrac User Transfer Subroutines Mass Transfer
Heat transfer coefficients	RadFrac User Transfer Subroutines Heat Transfer
Interfacial area	RadFrac User Transfer Subroutines Interfacial Area
Liquid and vapor holdup	RadFrac User Transfer Subroutines Holdup

RadFrac can also use subroutines to calculate KLL, tray or packing sizing/rating calculations, and reaction kinetics. These capabilities are not limited to rate-based mode. See chapters 11, 16, and 21 for details on these subroutines.

Rate-Based Distillation expects you to provide values for both vapor and liquid phases whenever you supply Fortran subroutines for:

- Binary mass transfer coefficients.
- Heat transfer coefficients.
- Interfacial area.
- Holdup.

¹ Aspen Rate-Based Distillation requires a separate license, and can be used only by customers who have licensed it through a specific license agreement with Aspen Technology, Inc.

Rate-Based Binary Mass Transfer Coefficient Subroutine

Calling Sequence for Binary Mass Transfer Coefficients

```
SUBROUTINE usrmtrfc†(KSTG, NCOMPS, IDX, NBOPST, KPDIAG, XCOMPB,
    FRATEL, YCOMPB, FRATEV, PRESS, TLIQ, TVAP,
    AVMWLI, AVMWVA, VISCML, DENMXL, SIGMAL,
    VISCMV, DENMXV, AREAIF, PREK, EXPKD, COLTYP,
    USRCOR, TWRARA, COLDIA, HTPACK, PACSIZ,
    SPAREA, CSIGMA, PFACT, PKPRMS, VOIDFR,
    IPAKAR, IPTYPE, IVENDR, IPMAT, IPSIZE,
    WEIRHT, DCAREA, ARAACT, FLOPTH, NPASS,
    WEIRL, IFMETH, SYSFAC, HOLEAR, ITTYPE,
    TRASPC, PITCH, IPHASE, NINT, INT, NREAL,
    REAL)
```

[†] Subroutine name you entered on the **RadFrac | User Transfer Subroutines | Mass Transfer** sheet.

Argument List for Binary Mass Transfer Coefficients

Variable	I/O [†]	Type	Dimension	Description and Range
KSTG	I	INTEGER	—	Stage number
NCOMPS	I	INTEGER	—	Number of components
IDX	I	INTEGER	NCOMPS	Component index vector
NBOPST	I	INTEGER	6	Physical property option set vector
KPDIAG	I	INTEGER	—	Physical property diagnostic code
XCOMPB	I	REAL*8	NCOMPS	Bulk liquid mole fraction
FRATEL	I	REAL*8	—	Flow of liquid (kgmole/s)
YCOMPB	I	REAL*8	NCOMPS	Bulk vapor mole fraction
FRATEV	I	REAL*8	—	Flow of vapor (kgmole/s)
PRESS	I	REAL*8	—	Pressure (N/m ²)
TLIQ	I	REAL*8	—	Liquid temperature (K)
TVAP	I	REAL*8	—	Vapor temperature (K)
AVMWLI	I	REAL*8	—	Average molecular weight of liquid mixture (kg/kgmole)
AVMWVA	I	REAL*8	—	Average molecular weight of vapor mixture (kg/kgmole)
VISCML	I	REAL*8	—	Viscosity of liquid (N-s/m ²)
DENMXL	I	REAL*8	—	Density of liquid mixture (kgmole/m ³)
SIGMAL	I	REAL*8	—	Surface tension of liquid (N/m)
VISCMV	I	REAL*8	—	Viscosity of vapor (N-s/m ²)
DENMXV	I	REAL*8	—	Density of vapor mixture (kgmole/m ³)

Variable	I/O [†]	Type	Dimension	Description and Range
AREAIF	I	REAL*8	—	Interfacial area When COLTYP = 1, AREAIF has units m ² /m ³ of packing volume When COLTYP = 2, AREAIF has units m ² /m ² of active tray area
PREK ^{††}	O	REAL*8	—	Mass transfer coefficient factor. See note.
EXPKD ^{††}	O	REAL*8	—	Mass transfer coefficient exponent. See note.
COLTYP	I	INTEGER	—	Type of column: 1 = Packed, 2 = Tray
USRCOR	I	INTEGER	—	Choice of user correlation
TWRARA	I	REAL*8	—	Cross-sectional area of tower (m ²)
COLDIA	I	REAL*8	—	Column diameter (m)
HTPACK	I	REAL*8	—	Height of packing in the section (m)
PACSI	I	REAL*8	—	Size of packing (m)
SPAREA	I	REAL*8	—	Specific surface area of packing (m ² /m ³)
CSIGMA	I	REAL*8	—	Critical surface tension of packing material (N/m)
PFACT	I	REAL*8	—	Packing factor (1/m)
PKPRMS	I	INTEGER	20	Packing parameters (see Packing Parameters)
VOIDFR	I	INTEGER	—	Void fraction of packing
IPAKAR	I	INTEGER	—	Packing arrangement: 1 = Random, 2 = Structured
IPTYPE	I	INTEGER	—	Packing Type (See Packing Type Specification)
IVENDR	I	INTEGER	—	Packing vendor code (See Packing Vendor Specification)
IPMAT	I	INTEGER	—	Packing material code (See Packing Material Specification)
IPSIZE	I	INTEGER	—	Packing size code (See Packing Size Specification)
WEIRHT	I	REAL*8	—	Height of exit weir (m)
DCAREA	I	REAL*8	—	Area of downcomer (m ²)
ARAACT	I	REAL*8	—	Active area available on tray (m ²)
FLOPTH	I	REAL*8	—	Flowpath length (m)
NPASS	I	INTEGER	—	Number of tray passes
WEIRL	I	REAL*8	—	Length of exit weir (m)
IFMETH	I	INTEGER	—	Flooding calculation method, required for sieve tray
SYSFAC	I	REAL*8	—	System factor, required for sieve tray
HOLEAR	I	REAL*8	—	Ratio of total hole area to active area, required for sieve tray
ITTYPE	I	INTEGER	—	Tray type: 1 = bubble cap, 2 = sieve, 3 = Glitsch Ballast, 4 = Koch Flexitray, 5 = Nutter Float Valve
TRASPC	I	REAL*8	—	Tray spacing (m)
PITCH	I	REAL*8	—	Sieve hole pitch (m), required for sieve trays
IPHASE	I	INTEGER	—	IPHASE: 0 = Liquid, 1 = Vapor

Variable	I/O [†]	Type	Dimension	Description and Range
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)

[†] I = Input to subroutine, O = Output from subroutine

^{††} $BINMTP = PREK \times DIFFUSIVITY^{EXPKD}$, where BINMTP is the binary mass transfer coefficients with molar density and interfacial area included, in the units of kgmole/s

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **RadFrac | User Transfer Subroutines | Mass Transfer** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Example: Binary Mass Transfer Coefficients Routine

You can use the following code as a template to create your own subroutine. In addition, Aspen Plus provides the template for this user subroutine online (see Appendix B).

```

      IMPLICIT NONE
      INTEGER KSTG, NCOMPS, IDX(NCOMPS), NBOPST(6), KPDIAG,
+          COLTYP, USRCOR, IPAKAR, IPTYPE, NPASS, IFMETH,
+          ITTYPE, NINT, INT(NINT), IPHASE, NREAL
      REAL*8 XCOMPB(NCOMPS), FRATEL, YCOMPB(NCOMPS), FRATEV,
+          PRESS, TLIQ, TVAP, AVMWLI, AVMWVA, VISCML, DENMXL,
+          SIGMAL, VISCML, DENMXV, AREAIF, PREK, EXPKD,
+          TWRARA, COLDIA, HTPACK, PACSIZ, SPAREA, CSIGMA,
+          PFACT, PKPRMS(20), VOIDFR, WEIRHT, DCAREA, ARAACT,
+          FLOPTH, WEIRL, SYSFAC, HOLEAR, TRASPC, PITCH,
+          REAL(NREAL)
C      Declare local variables used in the user correlations
C
      REAL*8 RS_BennettHL
      REAL*8 RS_BennettA
      REAL*8 RS_BennettC
      REAL*8 ScLB, ScVB, rhoLms, rhoVms, ReLPrm,
+          dTemp, uL, uV, Fs, QL,
+          C, alphae, hL, ShLB, ReV
C
C      Instead of computing BINMTP from diffusivity as in RATEFRAC
C      compute PREK and EXPKD for Rate-Based Distillation
C
```

```

      IF (COLTYP .EQ. 1) THEN
C
C***** PACKED COLUMN
C
      IF (USRCOR .EQ. 1) THEN
C
C        user subroutine example for packed column: Onda 68
C
C        Onda, K., Takeuchi, H. and Okumoto, Y., "Mass Transfer
C        Coefficients between Gas and Liquid Phases in Packed
C        Columns", J. Chem. Eng. Jap., 1, (1968) P56
C
C        IF (IPHASE.EQ.0) THEN
C
C          Liquid phase
C
C          rhoLms = DENMXL * AVMWLI
C          uL = FRATEL / TWRARA / DENMXL
C          ReLPrm = rhoLms * uL / VISCML / AREAIF
C          dTemp = (rhoLms/9.81D0/VISCML)**(0.33333333D0)
C          dTemp = 0.0051D0 * (ReLPrm**(0.66666667D0))
C          +          * ((SPAREA*PACSIZ)**(0.4D0)) / dTemp
C
C          CONVERT K FROM M/S TO KMOL/S
C          dTemp = dTemp * TWRARA * HTPACK * AREAIF * DENMXL
C
C          COMPOSITION INDEPENDENT PART OF SCHMIDT NUMBER
C          ScLB = VISCML / rhoLms
C
C          PREK = dTemp / DSQRT(ScLB)
C          EXPKD = 0.5D0
C
C        ELSE
C
C          Vapor phase
C
C          rhoVms = DENMXV * AVMWVA
C          uV = FRATEV / TWRARA / DENMXV
C          ReV = rhoVms * uV / VISC MV / SPAREA
C          dTemp = SPAREA*PACSIZ
C          dTemp = dTemp * dTemp
C          IF (PACSIZ .GE. 0.015D0) THEN
C            dTemp = 5.23D0 / dTemp
C          ELSE
C            dTemp = 2.0D0 / dTemp
C          END IF
C          dTemp = dTemp * (ReV**(0.7D0)) * SPAREA
C
C          CONVERT K FROM M/S TO KMOL/S
C          dTemp = dTemp * TWRARA * HTPACK * AREAIF * DENMXV
C
C          COMPOSITION INDEPENDENT PART OF SCHMIDT NUMBER
C          ScVB = VISC MV / rhoVms
C
C          PREK = dTemp * ScVB ** 0.33333333D0
C          EXPKD = 0.66666667D0
C        END IF
C      END OF IF (IPHASE)

```

```

C
C      END IF
C      END OF IF (USRCOR)
C
C      ELSE IF (COLTYP .EQ. 2) THEN
C
C**** TRAY COLUMN
C
C      IF (USRCOR .EQ. 1) THEN
C          user subroutine example for tray column: AIChE 58
C
C          AIChE, Bubble Tray Design Manual: Prediction of
C              Fractionation Efficiency, New York, 1958
C
C          For bubble cap, valve, and sieve trays
C
C          IF (IPHASE.EQ.0) THEN
C
C              Liquid phase
C
C              rhoVms = DENMXV * AVMWVA
C              rhoLms = DENMXL * AVMWLI
C              uV = FRATEV /DENMXV /ARAACT
C              Fs = uV * DSQRT(rhoVms)
C              C = RS_BennettC(WEIRHT)
C              QL = FRATEL/DENMXL
C              alphae = RS_BennettA(uV, rhoLms, rhoVms)
C              hL =RS_BennettHL (alphae, WEIRHT, C, QL, WEIRL)
C              dTemp = 19700.0D0 *(0.4D0*Fs+0.17D0) * hL
C          +
C              * ARAACT * DENMXL
C
C              PREK = dTemp
C              EXPKD = 0.5D0
C
C          ELSE
C              Vapor phase
C              rhoVms = DENMXV * AVMWVA
C              uV = FRATEV /DENMXV /ARAACT
C              Fs = uV * DSQRT(rhoVms)
C              QL = FRATEL/DENMXL
C              dTemp = 0.776 + 4.57*WEIRHT - 0.238*Fs
C          +
C              + 104.8*QL/WEIRL
C              dTemp = dTemp * uV * ARAACT * DENMXV
C
C              COMPOSITION INDEPENDENT PART OF SCHMIDT NUMBER
C              ScVB = VISC MV / rhoVms
C
C              PREK = dTemp /DSQRT(ScVB)
C              EXPKD = 0.5D0
C          END IF
C      END OF IF (IPHASE)
C      END IF
C      END OF IF (USRCOR)
C      END IF
C      END OF IF (COLTYP)
C      RETURN
C      END

```

Rate-Based Heat Transfer Coefficient Subroutine

Calling Sequence for Heat Transfer Coefficients

```
SUBROUTINE usrhtrfc†(KSTG, NCOMPS, IDX, NBOPST, KPDIAG, XCOMPB,
    FRATEL, YCOMPB, FRATEV, PRESS, TLIQ, TVAP,
    AVMWP, VISCMP, DENMXP, CPMIXP, THRMCP, PREH,
    EXPHK, EXPHD, COLTYP, USRCOR, COLDIA, IPAKAR,
    IPTYPE, IVENDR, IPMAT, IPSIZE, ITTYPE, IPHASE,
    NINT, INT, NREAL, REAL)
```

[†] Subroutine name you entered on the **RadFrac | User Transfer Subroutines | Heat Transfer** sheet.

Argument List for Heat Transfer Coefficients

Variable	I/O [†]	Type	Dimension	Description and Range
KSTG	I	INTEGER	—	Stage number
NCOMPS	I	INTEGER	—	Number of components
IDX	I	INTEGER	NCOMPS	Component index vector
NBOPST	I	INTEGER	6	Physical property option set vector
KPDIAG	I	INTEGER	—	Physical property diagnostic code
XCOMPB	I	REAL*8	NCOMPS	Bulk liquid mole fraction
FRATEL	I	REAL*8	—	Flow of liquid (kgmole/s)
YCOMPB	I	REAL*8	NCOMPS	Bulk vapor mole fraction
FRATEV	I	REAL*8	—	Flow of vapor (kgmole/s)
PRESS	I	REAL*8	—	Pressure (N/m ²)
TLIQ	I	REAL*8	—	Liquid temperature (K)
TVAP	I	REAL*8	—	Vapor temperature (K)
AVMWP	I	REAL*8	—	Average molecular weight of the mixture (kg/kgmole)
VISCMP	I	REAL*8	—	Viscosity of mixture (N-s/m ²)
DENMXP	I	REAL*8	—	Mixture density (kgmole/m ³)
CPMIXP	I	REAL*8	—	Mixture heat capacity (J/kgmole/K)
THRMCP	I	REAL*8	—	Mixture thermal conductivity (watt/m-K)
PREH ^{††}	O	REAL*8	—	Heat transfer coefficient factor (see note)
EXPHK ^{††}	O	REAL*8	—	Heat transfer coefficient exponent (see note)
EXPHD ^{††}	O	REAL*8	—	Heat transfer coefficient exponent (see note)
COLTYP	I	INTEGER	—	Type of column: 1 = Packed, 2 = Tray
USRCOR	I	INTEGER	—	Choice of user correlation
COLDIA	I	REAL*8	—	Column diameter (m)
IPAKAR	I	INTEGER	—	Packing arrangement: 1 = Random, 2 = Structured
IPTYPE	I	INTEGER	—	Packing Type (see Packing Type Specification)

Variable	I/O [†]	Type	Dimension	Description and Range
IVENDR	I	INTEGER	—	Packing vendor code (See Packing Vendor Specification)
IPMAT	I	INTEGER	—	Packing material code (See Packing Material Specification)
IPSIZE	I	INTEGER	—	Packing size code (See Packing Size Specification)
ITTYPE	I	INTEGER	—	Tray type: 1 = bubble cap, 2 = sieve, 3 = Glitsch Ballast, 4 = Koch Flexitray, 5 = Nutter Float Valve
IPHASE	I	INTEGER	—	Phase: 0 = Liquid, 1 = Vapor
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)

[†] I = Input to subroutine, O = Output from subroutine

$HTCOEF = PREH \times BINMTP^{EXPHK} \times DIFFUSIVITY^{EXPHD}$, where HTCOEF is the heat transfer coefficient with interfacial area included, in the units of J/sec-K

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **RadFrac | User Transfer Subroutines | Heat Transfer** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Example: Heat Transfer Coefficients Routine

You can use the following code as a template to create your own subroutine. In addition, Aspen Plus provides the template for this user subroutine online (see Appendix B).

```

      IMPLICIT NONE
      INTEGER KSTG, NCOMPS, IDX(NCOMPS), NBOPST(6), KPDIAG,
+         COLTYP, USRCOR, IPAKAR, IPTYPE, ITTYPE, IPHASE,
+         NINT, INT(NINT), NREAL
      REAL*8 XCOMPB(NCOMPS), FRATEL, YCOMPB(NCOMPS), FRATEV,
+         PRESS, TLIQ, TVAP, AVMWP, VISCMP, DENMXP, CPMIXP,
+         THRMCP, PREH, EXPHK, EXPHD, COLDIA, REAL(NREAL)

C      Declare local variables used in the user correlations
C
      REAL*8 ScOvPrB

C      Instead of computing HTCOEF for RATEFRAC
C      We will compute PREH, EXPHK, and EXPHD for
C      Rate-Based Distillation
C
      IF (USRCOR .EQ. 1) THEN
C      user subroutine example: Chilton-Colburn analogy
C
        IF (IPHASE.EQ.0) THEN
C
          Liquid phase
C
          ScOvPrB = THRMCP/(CPMIXP * DENMXP)
C
          PREH = CPMIXP * (ScOvPrB ** 0.666666667D0)
          EXPHK = 1.0D0
          EXPHD = -0.6666666667D0
        ELSE
C
          Vapor phase
C
          ScOvPrB = THRMCP/(CPMIXP * DENMXP)
C
          PREH = CPMIXP * (ScOvPrB ** 0.6666666667D0)
          EXPHK = 1.0D0
          EXPHD = -0.6666666667D0
        END IF
      END OF IF (IPHASE)
C
C
      END IF
      END OF IF (USRCOR)
C
      RETURN
      END

```

Rate-Based Interfacial Area Subroutine

Calling Sequence for Interfacial Area

```
SUBROUTINE usrintfa†(KSTG, NCOMPS, IDX, NBOPST, KPDIAG, XCOMPB,
    FRATEL, YCOMPB, FRATEV, PRESS, TLIQ, TVAP,
    AVMWLI, AVMWVA, VISCML, DENMXL, SIGMAL, VISCMV,
    DENMXV, AREAIF, COLTYP, USRCOR, TWRARA, COLDIA,
    HTPACK, PACSIZ, SPAREA, CSIGMA, PFACT, PKPRMS,
    VOIDFR, IPAKAR, IPTYPE, IVENDR, IPMAT, IPSIZE,
    WEIRHT, DCAREA, ARAACT, FLOPTH, NPASS, WEIRL,
    IFMETH, SYSFAC, HOLEAR, ITTYPE, TRASPC, PITCH,
    NINT, INT, NREAL, REAL)
```

[†] Subroutine name you entered on the **RadFrac | User Transfer Subroutines | Interfacial Area** sheet.

Argument List for Interfacial Area

Variable	I/O [†]	Type	Dimension	Description and Range
KSTG	I	INTEGER	—	Stage number
NCOMPS	I	INTEGER	—	Number of components
IDX	I	INTEGER	NCOMPS	Component index vector
NBOPST	I	INTEGER	6	Physical property option set vector
KPDIAG	I	INTEGER	—	Physical property diagnostic code
XCOMPB	I	REAL*8	NCOMPS	Bulk liquid mole fraction
FRATEL	I	REAL*8	—	Flow of liquid (kgmole/s)
YCOMPB	I	REAL*8	NCOMPS	Bulk vapor mole fraction
FRATEV	I	REAL*8	—	Flow of vapor (kgmole/s)
PRESS	I	REAL*8	—	Pressure (N/m ²)
TLIQ	I	REAL*8	—	Liquid temperature (K)
TVAP	I	REAL*8	—	Vapor temperature (K)
AVMWLI	I	REAL*8	—	Average molecular weight of liquid mixture (kg/kgmole)
AVMWVA	I	REAL*8	—	Average molecular weight of vapor mixture (kg/kgmole)
VISCML	I	REAL*8	—	Viscosity of liquid mixture (N-s/m ²)
DENMXL	I	REAL*8	—	Density of liquid mixture (kgmole/m ³)
SIGMAL	I	REAL*8	—	Surface tension of liquid (N/m)
VISCMV	I	REAL*8	—	Viscosity of vapor mixture (N-s/m ²)
DENMXV	I	REAL*8	—	Density of vapor mixture (kgmole/m ³)
AREAIF	O	REAL*8	—	Interfacial area. Units depend on COLTYP: When COLTYP = 1, m ² /m ³ of packing volume When COLTYP = 2, m ² /m ² of active tray area
COLTYP	I	INTEGER	—	Type of column: 1 = Packed, 2 = Tray
USRCOR	I	INTEGER	—	Choice of user correlation

Variable	I/O [†]	Type	Dimension	Description and Range
TWRARA	I	REAL*8	—	Cross-sectional area of tower (m ²)
COLDIA	I	REAL*8	—	Column diameter (m)
HTPACK	I	REAL*8	—	Height of packing in section (m)
PACSIK	I	REAL*8	—	Size of packing (m)
SPAREA	I	REAL*8	—	Specific surface area of packing (m ² /m ³)
CSIGMA	I	REAL*8	—	Critical surface tension of packing material (N/m)
PFACT	I	REAL*8	—	Packing factor (1/m)
PKPRMS	I	INTEGER 20		Packing parameters (See Packing Parameters)
VOIDFR	I	INTEGER	—	Void fraction of packing (m ³ /m ³)
IPAKAR	I	INTEGER	—	Packing arrangement: 1 = Random, 2 = Structured
IPTYPE	I	INTEGER	—	Packing Type (See Packing Type Specification)
IVENDR	I	INTEGER	—	Packing vendor code (See Packing Vendor Specification)
IPMAT	I	INTEGER	—	Packing material code (See Packing Material Specification)
IPSIZE	I	INTEGER	—	Packing size code (See Packing Size Specification)
WEIRHT	I	REAL*8	—	Height of exit weir (m)
DCAREA	I	REAL*8	—	Area of downcomer (m ²)
ARAACT	I	REAL*8	—	Active area available on tray (m ²)
FLOPTH	I	REAL*8	—	Flowpath length (m)
NPASS	I	INTEGER	—	Number of tray passes
WEIRL	I	REAL*8	—	Length of exit weir (m)
IFMETH	I	INTEGER	—	Flooding calculation method, required for sieve trays
SYSFAC	I	REAL*8	—	System factor, required for sieve trays
HOLEAR	I	REAL*8	—	Ratio of total hole area to active area, required for sieve tray
ITTYPE	I	INTEGER	—	Tray type: 1 = bubble cap, 2 = sieve, 3 = Glitsch Ballast, 4 = Koch Flexitray, 5 = Nutter Float Valve
TRASPC	I	REAL*8	—	Tray spacing (m)
PITCH	I	REAL*8	—	Sieve hole pitch (m), required for sieve trays
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER NINT		Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8 NREAL		Vector of real parameters (see Integer and Real Parameters)

[†] I = Input to subroutine, O = Output from subroutine

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the RadFrac | User Transfer Subroutines | Interfacial Area sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Example: Interfacial Area Routine

You can use the following code as a template to create you own subroutine. In addition, Aspen Plus provides the template for this user subroutine online (see Appendix B).

```

      IMPLICIT NONE
      INTEGER KSTG, NCOMPS, IDX(NCOMPS), NBOPST(6), KPDIAG,
+         COLTYP, USRCOR, IPAKAR, IPTYPE, NPASS, IFMETH,
+         ITTYPE, NINT, INT(NINT), NREAL
      REAL*8 XCOMPB(NCOMPS), FRATEL, YCOMPB(NCOMPS), FRATEV,
+         PRESS, TLIQ, TVAP, AVMWLI, AVMWVA, VISCML, DENMXL,
+         SIGMAL, VISCMV, DENMXV, AREAIF, TWRARA, COLDIA,
+         HTPACK, PACSIZ, SPAREA, CSIGMA, PFACT, PKPRMS(20),
+         VOIDFR, WEIRHT, DCAREA, ARAACT, FLOPTH, WEIRL,
+         SYSFAC, HOLEAR, TRASPC, PITCH, REAL(NREAL)
C      Declare local variables used in the user correlations
C
      REAL*8 WeL,    dTemp,  uV,    rhoVms,
+         uL,    rhoLms, ReL,    FrL,    uL2,
+         ReV,    d,        Wprime
C
C      Compute specific interface area as described above
C      Check COLTYP/USRCOR if providing multiple area correlations
C
      IF (COLTYP .EQ. 1) THEN
C
C**** PACKED COLUMN
C
      IF (USRCOR .EQ. 1) THEN
C          user subroutine example for packed column: Onda 68
C
C          Onda, K., Takeuchi, H. and Okumoto, Y., "Mass
Transfer
C          Coefficients between Gas and Liquid Phases in
Packed
C          Columns", J. Chem. Eng. Jap., 1, (1968) p. 56
C
      rhoLms = DENMXL * AVMWLI
      uL = FRATEL / TWRARA / DENMXL
      uL2 = uL * uL
      ReL = rhoLms * uL / VISCML / SPAREA
      FrL = SPAREA * uL2 / 9.81D0
C      WHERE 9.81D0 IS GRAVITY CONSTANT IN M/S^2
      WeL = rhoLms * uL2 / SIGMAL / SPAREA
      dTemp = -1.45D0*((CSIGMA/SIGMAL)**0.75D0)

```

```

+                               *(ReL**0.1D0)*(FrL**(-0.05D0))
+                               *(WeL**0.2D0)
      dTemp = 1.D0 - DEXP(dTemp)

      AREAIF = SPAREA*dTemp
      END IF
C      END OF IF (USRCOR)
C
      ELSE IF (COLTYP .EQ. 2) THEN
C
C**** TRAY COLUMN
C
      IF (USRCOR .EQ. 1) THEN
C      user subroutine example for tray column: Scheffe-
Weiland 87
C
C      Scheffe, R.D. and Weiland, R.H., "Mass Transfer
C      Characteristics of Valve Trays." Ind. Eng. Chem. Res.
C      26, (1987) p. 228
C
C      The original paper only mentioned valve tray.
C      It is also used for bubble-cap tray and sieve tray.
C
C      CHARACTERISTIC LENGTH IS ALWAYS 1 METER.
      d = 1.0D0
      rhoLms = DENMXL * AVMWLI
      rhoVms = DENMXV * AVMWVA
      uL = FRATEL / TWRARA / DENMXL
      uV = FRATEV / TWRARA / DENMXV
      ReL = rhoLms * uL * d / VISCML
      ReV = rhoVms * uV * d / VISCMV
      Wprime = WEIRHT / d
      AREAIF = 0.27D0 * ReV**0.375D0 * ReL**0.247D0
      AREAIF = AREAIF * Wprime**0.515
      END IF
C      END OF IF (USRCOR)
C
      END IF
C      END OF IF (COLTYP)
C
      RETURN
      END

```

Rate-Based Holdup Subroutine

Calling Sequence for Holdup Subroutine

```
SUBROUTINE usrhldup†(KSTG, FRATEL, FRATEV, AVMWLI, AVMWVA, VISCML,
DENMXL, SIGMAL, VISC MV, DENMXV, LHLDUP, VHLDUP,
VSPACE, COLTYP, USRCOR, TWRARA, COLDIA, HTPACK,
PACSIZ, SPAREA, CSIGMA, PFACT, PKPRMS, VOIDFR,
PLHOLD, PVHOLD, IPAKAR, IPTYPE, IVENDR, IPMAT,
IPSIZE, WEIRHT, DCAREA, ARAACT, FLOPTH, NPASS,
WEIRL, IFMETH, SYSFAC, HOLEAR, ITTYPE, TRASPC,
PITCH, NINT, INT, NREAL, REAL)
```

[†] Subroutine name you entered on the **RadFrac | User Transfer Subroutines | Holdup** sheet.

Argument List for Holdup Subroutine

Variable	I/O [†]	Type	Dimension	Description and Range
KSTG	I	INTEGER	—	Stage number
FRATEL	I	REAL*8	—	Flow of liquid (kgmole/s)
FRATEV	I	REAL*8	—	Flow of vapor (kgmole/s)
AVMWLI	I	REAL*8	—	Average molecular weight of liquid mixture (kg/kgmole)
AVMWVA	I	REAL*8	—	Average molecular weight of vapor mixture (kg/kgmole)
VISCML	I	REAL*8	—	Viscosity of liquid (N-s/m ²)
DENMXL	I	REAL*8	—	Density of liquid mixture (kgmole/ m ³)
SIGMAL	I	REAL*8	—	Surface tension of liquid (N/m)
VISC MV	I	REAL*8	—	Viscosity of vapor mixture (N-s/m ²)
DENMXV	I	REAL*8	—	Density of vapor mixture (kgmole/m ³)
LHLDUP	O	REAL*8	—	Liquid stage holdup (m ³)
VHLDUP	O	REAL*8	—	Vapor stage holdup (m ³)
VSPACE	O	REAL*8	—	Vapor space holdup (m ³)
COLTYP	I	INTEGER	—	Type of column: 1 = Packed, 2 = Tray
USRCOR	I	INTEGER	—	Choice of user correlation
TWRARA	I	REAL*8	—	Cross-sectional area of tower (m ²)
COLDIA	I	REAL*8	—	Column diameter (m)
HTPACK	I	REAL*8	—	Height of packing in the section (m)
PACSIZ	I	REAL*8	—	Size of packing (m)
SPAREA	I	REAL*8	—	Specific surface area of packing (m ² /m ³)
CSIGMA	I	REAL*8	—	Critical surface tension of packing material (N/m)
PFACT	I	REAL*8	—	Packing factor (1/m)
PKPRMS	I	INTEGER	20	Packing parameters (See Packing Parameters)
VOIDFR	I	INTEGER	—	Void fraction of packing (m ³ /m ³)

Variable	I/O [†]	Type	Dimension	Description and Range
PLHOLD	I	REAL*8	—	User specified % free volume for liquid holdup
PVHOLD	I	REAL*8	—	User specified % free volume for vapor holdup
IPAKAR	I	INTEGER	—	Packing arrangement: 1 = Random, 2 = Structured
IPTYPE	I	INTEGER	—	Packing Type (see Packing Type Specification)
IVENDR	I	INTEGER	—	Packing vendor code (See Packing Vendor Specification)
IPMAT	I	INTEGER	—	Packing material code (See Packing Material Specification)
IPSIZE	I	INTEGER	—	Packing size code (See Packing Size Specification)
WEIRHT	I	REAL*8	—	Height of exit weir (m)
DCAREA	I	REAL*8	—	Area of downcomer (m ²)
ARAACT	I	REAL*8	—	Active area available on tray (m ²)
FLOPTH	I	REAL*8	—	Flowpath length (m)
NPASS	I	INTEGER	—	Number of tray passes
WEIRL	I	REAL*8	—	Length of exit weir (m)
IFMETH	I	INTEGER	—	Flooding calculation method, required for sieve trays
SYSFAC	I	REAL*8	—	System factor, required for sieve trays
HOLEAR	I	REAL*8	—	Ratio of total hole area to active area, required for sieve tray
ITTYPE	I	INTEGER	—	Tray type: 1 = bubble cap, 2 = sieve, 3 = Glitsch Ballast, 4 = Koch Flexitray, 5 = Nutter Float Valve
TRASPC	I	REAL*8	—	Tray spacing (m)
PITCH	I	REAL*8	—	Sieve hole pitch (m), required for sieve trays
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)

[†] I = Input to subroutine, O = Output from subroutine

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the RadFrac | User Transfer Subroutines | Holdup sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Example: Holdup Subroutine

You can use the following code as a template to create your own subroutine. In addition, Aspen Plus provides the template for this user subroutine online (see Appendix B).

```

      IMPLICIT NONE
      INTEGER KSTG, COLTYP, USRCOR, IPAKAR, IPTYPE, NPASS,
1      IFMETH, ITTYPE, NINT, INT(NINT), NREAL
      REAL*8 FRATEL, FRATEV, AVMWLI, AVMWVA, VISCML, DENMXL,
1      SIGMAL, VISCMLV, DENMXV, LHLDP, VHLDP, VSPACE,
2      TWRARA, COLDIA, HTPACK, PACSIZ, SPAREA, CSIGMA,
3      PFACT, PKPRMS(20), VOIDFR, PLHOLD, PVHOLD, WEIRHT,
4      DCAREA, ARAACT, FLOPTH, WEIRL, SYSFAC, HOLEAR,
5      TRASPC, PITCH, REAL(NREAL)

C
C      Define local variables
C
      INTEGER ITER, KHTERR, KDPERR
C      Variables used in the Stichlmair 89 correlation
      REAL*8 DEQ, UL, UV, REV, C1, C2, C3,
+      DP, DDPDRY, DPWET, FRL, HT, HT0, AUX, F, D,
+      C_S, GRAV, FF, HTETA
C
C      Variables used in the Bennett 83 correlation
      REAL*8 RS_BennettA, RS_BennettC, RS_BennettHL
      REAL*8 FREVOL, US, RHOV, RHOL, ALPHAE, C_B, QL, HL, HF,
+      VOID, PLH, PVH
      DATA GRAV /9.806599D0/
C
      IF (COLTYP .EQ. 1) THEN
C
C      PACKED COLUMN
C
          VSPACE = 0.0D0
          IF (USRCOR .EQ. 1) THEN
C      user subroutine example for packed column: Stichlmair 89
C
C      Stichlmair, J., Bravo, J.L. and Fair, J.R., "General Model
C      for Prediction of Pressure Drop and Capacity of
C      Countercurrent Gas/Liquid Packed Columns", Gas Sep.
C      Purif., 3, (1989), P19
C
          DEQ = 6D0*(1D0 - VOIDFR)/SPAREA
          RHOL = AVMWLI*DENMXL
          RHOV = AVMWVA*DENMXV
C
C      ***      CALCULATE FRICTION FACTOR      ***
C
          UV = FRATEV/DENMXV/TWRARA
          REV = DEQ*UV*RHOV/VISCMLV
          C1 = PKPRMS(1)
          C2 = PKPRMS(2)
          C3 = PKPRMS(3)
          FF = C1/REV + C2/DSQRT(REV) + C3
          IF (FF .EQ. 0D0) FF = 10D0
          C_S = (-C1/REV - C2/2D0/DSQRT(REV))/FF

```



```

C
C ***      CALCULATE DRY PRESSURE DROP ***
C
      DPDRY = 0.75D0*FF*(1D0 - VOIDFR)/VOIDFR**4.65D0
1          *RHOV*UV*UV/DEQ/RHOL/GRAV
C
C ***      CALCULATE LIQUID HOLDUP BELOW THE LOADING POINT ***
C
      UL = FRATEL/DENMXL/TWRARA
      FRL = UL*UL*SPAREA/GRAV/VOIDFR**4.65D0
      HT0 = .555D0*FRL**(0.33333333D0)
C
C ***      SET INITIAL ESTIMATE OF WET PRESSURE DROP ***
C
      DP = DPDRY
      ITER = 0
C
C ***      CALCULATE WET PRESSURE DROP USING NEWTON'S METHOD ***
C
101      KHTERR = 0
      HT = HT0*(1D0 + 20D0*DP*DP)
      HTETA = HT/VOIDFR
      IF (HTETA .GE. 1D0) THEN
        KHTERR = 1
      ELSE
        AUX = ((1D0 -VOIDFR*(1D0 -HTETA))/(1D0 -VOIDFR))
1          **((2D0 +C_S)/3D0)
        F = DP/DPDRY -AUX/(1D0 -HTETA)**4.65D0
        D = 1D0/DPDRY -40D0*HT0*DP*AUX/(1D0 -HTETA)**
1          4.65D0*(4.65/(VOIDFR -HT) +
2          (2D0+C_S)/3D0/(1D0 -VOIDFR +HT))
      END IF
      END OF IF (HTETA...)
C
C ***      CHECK IF LIQUID OCCUPIES THE WHOLE PACKING VOIDAGE ***
C
      IF (KHTERR .EQ. 1) THEN
        HT = DMAX1(VOIDFR, HT0)
        DPWET = DSQRT((HT/HT0 - 1D0)/20D0)
        GOTO 301
      END IF
      END OF IF (KHTERR...)
C
C ***      GET NEW ESTIMATE ***
C
      DPWET = DP - F/D
C
C ***      CHECK FOR CONVERGENCE ***
C
      IF (DABS(DPWET - DP)/DP .GT. 1D-12) THEN
        IF (DPWET .GT. 0.3D0) DPWET = 0.3D0
        IF (DPWET .LT. 0.0D0) DPWET = 0.01D0
        ITER = ITER + 1
        IF (ITER .GT. 30) THEN
          KDPERR = 5
          GOTO 201
        END IF
      END IF

```

```

        DP = DPWET
        GOTO 101
    END IF
C      END OF IF (DABS...
C
C ***      CALCULATE TOTAL LIQUID HOLDUP ***
C
201      HT = HT0 * (1D0 + 20D0*DPWET*DPWET)
301      LHL DUP = HT * TWRARA * HTPACK
        VHLDUP = (1D0 - HT - VOIDFR) * TWRARA * HTPACK
C
        END IF
C      END OF IF (USRCOR...
C
        ELSE IF (COLTYP .EQ. 2) THEN
C
C      TRAY COLUMN
C
        IF (USRCOR .EQ. 1) THEN
C      user subroutine example for tray column: Bennett 83
C
C      Bennett, D.L., Agrawal, R. and Cook, P.J., "New Pressure
C      Drop Correlation for Sieve Tray Distillation Columns",
C      AIChE J., 29, (1983) p 434
C
        US = FRATEV/DENMXV/ARAACT
        RHOV = DENMXV * AVMWVA
        RHOL = DENMXL * AVMWLI
        ALPHA E = RS_BennettA(US, RHOL, RHOV)
        C_B = RS_BennettC(WEIRHT)
        QL = FRATEL/DENMXL
        HL =RS_BennettHL (ALPHA E, WEIRHT, C_B, QL, WEIRL)
        HF = HL/ALPHA E
        LHL DUP = HL*ARAACT
        VHLDUP = (HF-HL)*ARAACT
        VSPACE = TRASPC*(ARAACT+DCAREA)
+      - (LHL DUP+VHLDUP)*(ARAACT+DCAREA)/ARAACT
        END IF
C      END OF IF (USRCOR...
C      END IF
C      END OF IF (COLTYP...
        RETURN
    END

```

Packing Parameters

The vector PKPRMS contains the following values:

Element	Description
1	Stichlmair constant C1
2	Stichlmair constant C2
3	Stichlmair constant C3
4	CL in Billet 1993 correlation
5	CV in Billet 1993 correlation

Element	Description
6	B in Bravo-Rocha-Fair 1985 correlation
7	S in Bravo-Rocha-Fair 1985 correlation
8	H in Bravo-Rocha-Fair 1985 correlation
9	F_{se} in Bravo-Rocha-Fair 1992 correlation
10	CE in Bravo-Rocha-Fair 1992 correlation
11	θ in Bravo-Rocha-Fair 1992 correlation

See the reference for RadFrac in the online help for detailed descriptions of these correlations and parameters.

Packing Type Specification

The following list provides the packing type specification for the variable IPTYPE.

Random Packings	Structured Packings
1 = Berl saddle	100 = Glitsch Grid
2 = Cascade mini-ring (CMR)	200 = Glitsch Goodloe
3 = Coil Pack	501 = Mellapak 125X
4 = Dixon Packing	502 = Mellapak 125Y
5 = Heli Pack	503 = Mellapak 250X
6 = Helix	504 = Mellapak 250Y
7 = Hy-Pak	505 = Mellapak 350X
8 = I-ball packing	506 = Mellapak 350Y
9 = Intalox metal tower packing (IMTP)	507 = Mellapak 500X
10 = Intalox saddle	508 = Mellapak 500Y
11 = Super-Intalox saddle	509 = Sulzer BX
12 = Leschig ring	511 = Sulzer CY
13 = McMahon packing	512 = Kerapak
14 = Mesh ring packing	513 = Mellapak 170Y
15 = Pall ring	607 = Norton Intalox structured packing (ISP)
16 = Raschig ring	701 = Koch Flexipac
17 = Sigma packing	702 = Koch Flexeramic
18 = Intalox Snowflake plastic packing	703 = Koch Flexigrid
19 = Koch Flexiring single-tab slotted ring	801 = Raschig Ralu-pak
20 = Koch HCKP multi-tab slotted ring	802 = Raschig Super-Pak
21 = Koch Fleximax	803 = Crossflgrd
22 = Koch Flexisaddle	
23 = Raschig Ralu-ring	
24 = Raschig Ralu-flow	
25 = Raschig Super-Ring	
26 = Raschig Torus-Saddle	
27 = Raschig Super-Torus-Saddle	

If the packing type is not specified, IPTYPE = USER_IUMISS.

Packing Vendor Specification

The following list provides the packing type specification for the variable IVENDR.

3 = MTL	6 = GENERIC	8 = SULZER
4 = NORTON	7 = KOCH	9 = RASCHIG

If the packing vendor is not specified, IVENDR = USER_IUMISS.

Packing Material Specification

The following list provides the packing type specification for the variable IPMAT.

1 = PLASTIC	4 = CARBON	7 = STANDARD
2 = METAL	5 = METAL-32	8 = STEEL
3 = CERAMIC	6 = METAL-16	9 = POLYPROPYLENE

If the packing material is not specified, IPMAT = USER_IUMISS.

Packing Size Specification

The following list provides the packing type specification for the variable IPSIZE.

101 = 1.5-MM	205 = NO-0.5P	409 = 50
102 = 2-MM	206 = NO-0.5	501 = STYLE-2
103 = 2.5-MM	207 = NO-1A	502 = STYLE-3
104 = 3-MM	208 = NO-1P	601 = 1X
105 = 4-MM	209 = NO-1X	602 = 1Y
106 = 5-MM	210 = NO-1	603 = 1.4Y
107 = 6-MM 0.25"	211 = NO-1.5P	604 = 1.6Y
108 = 10-MM 0.375"	212 = NO-2A	605 = 2X
109 = 13-MM 0.5"	213 = NO-2P	606 = 2Y
110 = 15-MM 0.6"	214 = NO-2X	607 = 2.5Y
111 = 16-MM 0.625"	215 = NO-2	608 = 3X
112 = 19-MM 0.75"	216 = NO-3A	609 = 3Y
113 = 20-MM 0.875"	217 = NO-3P	610 = 4Y
114 = 25-MM 1"	218 = NO-3	611 = 125X
115 = 30-MM 1.25"	219 = NO-4P	612 = 125Y
116 = 35-MM 1.375"	220 = NO-4	613 = 250X
117 = 38-MM 1.5"	221 = NO-5P	614 = 250Y
118 = 40-MM 1.6"	222 = NO-5	615 = 350X
119 = 50-MM 2"	223 = NO-7	616 = 350Y
120 = 75-MM 3"	224 = NO-0.3	617 = 500X
121 = 90-MM 3.5"	225 = NO-0.7	618 = 500Y
122 = 100-MM 4"	226 = NO-1.5301 - II	619 = 170Y
123 = 45-MM 1.8"	401 = 200	627 = 250YC
124 = 60-MM 2.36"	402 = 300	701 = 1T
127 = 80-MM	403 = 400	702 = 2T
128 = 125-MM	404 = 700	703 = 3T
201 = NO-0A	405 = 28	704 = 4T
202 = NO-0P	406 = 48	705 = 5T
203 = NO-0	407 = 88	
204 = NO-0.5A	408 = 25	

If the packing size is not specified, IPSIZE = USER_IUMISS.

21 User Tray and Packing Subroutines

You can supply your own tray sizing or packing subroutine for sizing and rating calculations for RadFrac, MultiFrac, and PetroFrac.

For trays, enter the subroutine name on the following sheet:

For	Use this sheet
RadFrac	RadFrac User Subroutines Tray Sizing/Rating
MultiFrac	MultiFrac User Subroutines Tray Sizing/Rating
PetroFrac	PetroFrac User Subroutines Tray Sizing/Rating

For packing, enter the subroutine name on the following sheet:

For	Use this sheet
RadFrac	RadFrac User Subroutines Pack Sizing/Rating
MultiFrac	MultiFrac User Subroutines Pack Sizing/Rating
PetroFrac	PetroFrac User Subroutines Pack Sizing/Rating

User Tray Sizing/Rating Subroutine

Calling Sequence for User Tray Subroutine

SUBROUTINE subrname[†] (MODE, N, FLM, FVMTO, FLV, FVVTO, RHOL, RHOVTO, SIGMA, FP, QR, SYSFAC, NPASS, TS, FA, DIAM, SZPAR, RTPAR, NINT, INT, NREAL, REAL, SZRSLT, RTRSLT, P, XMWL, XMWVTO, XMUL, XMUVTO, FFAC, FFR, ITTYPE)

[†]Subroutine name you entered on the **User Subroutines | Tray Sizing/Rating** sheet.

Argument List Descriptions for User Tray Subroutine

Variable	I/O [†]	Type	Dimension	Description and Range
MODE	I	INTEGER	—	Calculation mode: 1 = Sizing, 2 = Rating
N	I	INTEGER	—	Stage number
FLM	I	REAL*8	—	Mass liquid flow from the stage (kg/s)
FVMTO	I	REAL*8	—	Mass vapor flow to the stage (kg/s)
FLV	I	REAL*8	—	Volumetric liquid flow from the stage (m ³ /s)
FVVTO	I	REAL*8	—	Volumetric vapor flow to the stage (m ³ /s)
RHOL	I	REAL*8	—	Mass density of liquid from the stage (kg/m ³)
RHOVTO	I	REAL*8	—	Mass density of vapor to the stage (kg/m ³)
SIGMA	I	REAL*8	—	Surface tension of liquid from the stage (N/m)
FP	I	REAL*8	—	Flow parameter
QR	I	REAL*8	—	Reduced vapor throughput (m ³ /s)
SYSFAC	I	REAL*8	—	System foaming factor
NPASS	I	INTEGER	—	Number of passes
TS	I	REAL*8	—	Tray spacing (m)
FA	I/O	REAL*8	—	Percentage approach to flooding Input for sizing, Output for rating
DIAM	I/O	REAL*8	—	Column diameter (m) Input for rating, Output for sizing
SZPAR	I	REAL*8	3	Sizing input parameters: SZPAR (1) - minimum allowed downcomer area/tray area SZPAR (2) - percent slot or hole area/active area SZPAR (3) - minimum column diameter (m)
RTPAR	I	REAL*8	60	Rating input parameters (see RTPAR)
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)

Variable	I/O†	Type	Dimension	Description and Range
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
SZRSLT	O	REAL*8	4	Sizing results: SZRSLT (1) - Weir length (m) SZRSLT (2) - Downcomer velocity (m/s) SZRSLT (3) - Downcomer area per panel (m ²) SZRSLT (4) - Active area (m ²)
RTRSLT	O	REAL*8	20	Rating results (see RTRSLT)
P	I	REAL*8	—	Stage pressure (N/m ²)
XMWL	I	REAL*8	—	Average molecular weight of liquid from the stage
XMWVTO	I	REAL*8	—	Average molecular weight of vapor to the stage
XMULI	I	REAL*8	—	Viscosity of liquid from stage (N-s/m ²)
XMOVTO	I	REAL*8	—	Viscosity of vapor to the stage (N-s/m ²)
FFAC	I	REAL*8	—	F factor (for rating only) (m/s (kg/m ³) ^{1/2})
FFR	I	REAL*8	—	Reduced F factor (m ³ /s (kg/m ³) ^{1/2})
ITTYPE	I	INTEGER	—	Type of tray 1 = bubble-cap 2 = sieve

† I = Input to subroutine, O = Output from subroutine

RTPAR

The array of rating input parameters contains these values in its 60 elements:

RTPAR (1) - Weir height (m), panel A	RTPAR (31) - Downcomer height (m), panel C
RTPAR (2) - Weir height (m), panel B	RTPAR (32) - Downcomer height (m), panel D
RTPAR (3) - Weir height (m), panel C	RTPAR (33) - Flow path length (m), panel A
RTPAR (4) - Weir height (m), panel D	RTPAR (34) - Flow path length (m), panel B
RTPAR (5) - Weir length (m), panel A	RTPAR (35) - Flow path length (m), panel C
RTPAR (6) - Weir length (m), panel B	RTPAR (36) - Flow path length (m), panel D
RTPAR (7) - Weir length (m), panel C	RTPAR (37) - Flow path width (m), panel A
RTPAR (8) - Weir length (m), panel D	RTPAR (38) - Flow path width (m), panel B
RTPAR (9) - Top downcomer width (m), panel A	RTPAR (39) - Flow path width (m), panel C
RTPAR (10) - Top downcomer width (m), panel B	RTPAR (40) - Flow path width (m), panel D
RTPAR (11) - Top downcomer width (m), panel C	RTPAR (41) - Active area (m ²), panel A
RTPAR (12) - Top downcomer width (m), panel D	RTPAR (42) - Active area (m ²), panel B
RTPAR (13) - Bottom downcomer width (m), panel A	RTPAR (43) - Active area (m ²), panel C
RTPAR (14) - Bottom downcomer width (m), panel B	RTPAR (44) - Active area (m ²), panel D
RTPAR (15) - Bottom downcomer width (m), panel C	RTPAR (45) - Number of caps, panel A
RTPAR (16) - Bottom downcomer width (m), panel D	RTPAR (46) - Number of caps, panel B
RTPAR (17) - Top downcomer area (m ²), panel A	RTPAR (47) - Number of caps, panel C
RTPAR (18) - Top downcomer area (m ²), panel B	RTPAR (48) - Number of caps, panel D
RTPAR (19) - Top downcomer area (m ²), panel C	RTPAR (49) - Number of rows, panel A
RTPAR (20) - Top downcomer area (m ²), panel D	RTPAR (50) - Number of rows, panel B
RTPAR (21) - Bottom downcomer area (m ²), panel A	RTPAR (51) - Number of rows, panel C
RTPAR (22) - Bottom downcomer area (m ²), panel B	RTPAR (52) - Number of rows, panel D
RTPAR (23) - Bottom downcomer area (m ²), panel C	RTPAR (53) - Downcomer-to-wall (m)
RTPAR (24) - Bottom downcomer area (m ²), panel D	RTPAR (54) - Hole diameter (m)
RTPAR (25) - Downcomer clearance (m), panel A	RTPAR (55) - Hole area/active area
RTPAR (26) - Downcomer clearance (m), panel B	RTPAR (56) - Deck thickness (m)
RTPAR (27) - Downcomer clearance (m), panel C	RTPAR (57) - Cap diameter (m)
RTPAR (28) - Downcomer clearance (m), panel D	RTPAR (58) - Cap space (m)
RTPAR (29) - Downcomer height (m), panel A	RTPAR (59) - Skirt height (m)
RTPAR (30) - Downcomer height (m), panel B	RTPAR (60) - Efficiency

RTRSLT

The rating results array contains these parameters:

RTRSLT (1) - Pressure drop (mm - liq), panel A	RTRSLT (13) - Downcomer backup/tray spacing, panel A
RTRSLT (2) - Pressure drop (mm - liq), panel B	RTRSLT (14) - Downcomer backup/tray spacing, panel B
RTRSLT (3) - Pressure drop (mm - liq), panel C	RTRSLT (15) - Downcomer backup/tray spacing, panel C
RTRSLT (4) - Pressure drop (mm - liq), panel D	RTRSLT (16) - Downcomer backup/tray spacing, panel D
RTRSLT (5) - Downcomer velocity (gpm/ft ²), panel A	RTRSLT (17) - Weir loading (gpm/ft), panel A
RTRSLT (6) - Downcomer velocity (gpm/ft ²), panel B	RTRSLT (18) - Weir loading (gpm/ft), panel B
RTRSLT (7) - Downcomer velocity (gpm/ft ²), panel C	RTRSLT (19) - Weir loading (gpm/ft), panel C
RTRSLT (8) - Downcomer velocity (gpm/ft ²), panel D	RTRSLT (20) - Weir loading (gpm/ft), panel D
RTRSLT (9) - Downcomer backup (mm), panel A	
RTRSLT (10) - Downcomer backup (mm), panel B	
RTRSLT (11) - Downcomer backup (mm), panel C	
RTRSLT (12) - Downcomer backup (mm), panel D	

User Packing Sizing/Rating Subroutine

Calling Sequence for User Packing Subroutine

SUBROUTINE subrname[†] (MODE, N, P, FLM, FVMTO, FLV, FVVTO, XMWL, XMWVTO, RHOL, RHOVTO, XMUL, XMUVTO, SIGMA, FP, QR, FFAC, FFR, SYSFAC, IPTYPE, IPSIZE, IPMAT, PACKFC, VOID, SURFA, STICH, HETP, FA, DIAM, NINT, INT, NREAL, REAL, DPSTG, HTSTG)

[†] Subroutine name you entered on the **User Subroutines | Pack Sizing/Rating** sheet.

Argument List Descriptions for User Packing Subroutine

Variable	I/O [†]	Type	Dimension	Description and Range
MODE	I	INTEGER	—	Calculation mode: 1 = Sizing, 2 = Rating
N	I	INTEGER	—	Stage number
P	I	REAL*8	—	Stage pressure (N/m ²)
FLM	I	REAL*8	—	Mass liquid flow from the stage (kg/s)
FVMTO	I	REAL*8	—	Mass vapor flow to the stage (kg/s)
FLV	I	REAL*8	—	Volumetric liquid flow from the stage (m ³ /s)
FVVTO	I	REAL*8	—	Volumetric vapor flow to the stage (m ³ /s)
XMWL	I	REAL*8	—	Average molecular weight of liquid from the stage
XMWVTO	I	REAL*8	—	Average molecular weight of vapor to the stage
RHOL	I	REAL*8	—	Mass density of liquid from the stage (kg/m ³)
RHOVTO	I	REAL*8	—	Mass density of vapor to the stage (kg/m ³)
XMUL	I	REAL*8	—	Viscosity of liquid from the stage (N-s/m ²)
XMUVTO	I	REAL*8	—	Viscosity of vapor to the stage (N-s/m ²)
SIGMA	I	REAL*8	—	Surface tension of liquid from the stage (N/m)
FP	I	REAL*8	—	Flow parameter
QR	I	REAL*8	—	Reduced vapor throughput (m ³ s)
FFAC	I	REAL*8	—	F factor (for rating only) (m/s (kg/m ³) ^{1/2})
FFR	I	REAL*8	—	Reduced F factor (m/s (kg/m ³) ^{1/2})
SYSFAC	I	REAL*8	—	System foaming factor
IPTYPE	I	INTEGER	—	Packing type
IPSIZE	I	INTEGER	—	Packing size
IPMAT	I	INTEGER	—	Packing material
PACKFC	I	REAL*8	—	Packing factor (1/m)
VOID	I	REAL*8	—	Packing void fraction
SURFA	I	REAL*8	—	Packing surface area (m ² /m ³)
STICH	I	REAL*8	3	Stichlmair constants

Variable	I/O [†]	Type	Dimension	Description and Range
HETP	I/O	REAL*8	—	Height equivalent to a theoretical plate (m)
FA	I/O	REAL*8	—	Fractional approach to flooding Input for sizing, Output for rating
DIAM	I/O	REAL*8	—	Column diameter (m) Input for sizing, Output for rating
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I/O	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	REAL*8	—	Number of real parameters (see Integer and Real Parameters)
REAL	I/O	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
DPTSG	O	REAL*8	—	Pressure drop per stage (N/m ²)
HTSTG	O	REAL*8	—	Fractional liquid holdup

[†] I = Input to subroutine, O = Output from subroutine

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on one of the sheets listed on page 243. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

22 User Performance Curves Subroutine for Compr/MCompr

This chapter describes the user performance curves subroutine for Compr and MCompr.

To supply a user performance curves subroutine for Compr, select **User Subroutine** for **Curve Format** on the **Compr | Performance Curves | Curve Setup** sheet. On the **Compr | User Subroutine | Specification** sheet, choose the type of user curves and specify the user subroutine name.

To supply a user performance curves subroutine for MCompr, select **User Subroutine** for **Curve Format** on the **MCompr | Performance Curves | Curve Setup** sheet. On the **MCompr | User Subroutine | Specification** sheet, choose the type of user curves and specify the user subroutine name.

The user subroutine calculates one of the following:

- Head.
- Head coefficient.
- Power.
- Discharge pressure.
- Pressure ratio.
- Pressure change.

In addition, you can also calculate efficiency in the user subroutine.

Performance Curve Subroutine

Calling Sequence for User Performance Curves Subroutine

SUBROUTINE subrname[†] (S1, NSUBS, IDXSUB, ITYPE, NBOPST, NINT, INT, NREAL, REAL, NIWORK, IWORK, NWORK, WORK, IEXTOP, ISTAGE, IWHEEL, DIAMIM, ACTSPD, SONCVI, IUSRCT, VALD1U, VALD2U, ABVSRG, BELSWL, IRETFL)

[†] Subroutine name you entered on the **Compr** or **MCompr | User Subroutine | Specification** sheet.

Argument List for User Performance Curves Subroutine

Variable	I/O [†]	Type	Dimension	Description and Range
S1	I	REAL*8	(1)	Stream vector after initial flash (has already accounted for pressure losses at suction nozzle)
NSUBS	I	INTEGER	—	Number of substreams
IDXSUB	I	INTEGER	NSUBS	Substream index vector
ITYPE	I	INTEGER	NSUBS	Substream type vector: 1=MIXED, 2=CISOLID, 3=NC
NBOPST	I	INTEGER	6	Property option set
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
NIWORK	I	INTEGER	—	Length of user integer work array (see Local Work Arrays)
IWORK	I	INTEGER	NIWORK	User integer work array (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of user real work array (see Local Work Arrays)
WORK	I	REAL*8	NWORK	User real work area (see Local Work Arrays)
IEXTOP	I	INTEGER	—	Extrapolation option = 1, extrapolate beyond surge and stonewall = 0, do not extrapolate
ACTSPD	I	REAL*8	—	Actual operating shaft speed (1/sec)
ISTAGE	I	INTEGER	—	Stage number (= 1 for Compr)
IWHEEL	I	INTEGER	—	Wheel number (= 1 for Compr)
DIAMIM	I	REAL*8	—	Impeller diameter (m)
SONCVI	I	REAL*8	—	Sonic velocity of process fluid at suction conditions (after any suction pressure losses)

Variable	I/O [†]	Type	Dimension	Description and Range
IUSRCT	I	INTEGER	—	Type of dependent variable (specified on the User Subroutine Specification sheet) 0=none 1=head 2=head coefficient 3=power 4=discharge pressure 5=pressure ratio 6=pressure change
VALD1U	O	REAL*8	—	Calculated value of dependent variable (in SI units)
VALD2U	O	REAL*8	—	Calculated value of efficiency
ABVSRG	O	REAL*8	—	Percentage above surge point $100 * (\text{actual suction volumetric flowrate} - \text{suction volumetric flowrate at surge}) / \text{suction volumetric flowrate at surge}$
BELSWL	O	REAL*8	—	Percentage below stonewall point $100 * (\text{suction volumetric flowrate at stonewall} - \text{actual suction volumetric flowrate}) / \text{suction volumetric flowrate at stonewall}$
IRETFL	O	INTEGER	—	Return flag = 0, all fine > 0, error = -1, value below surge = -2, value above stonewall

[†] I = Input to subroutine, O = Output from subroutine

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **Compr** or **MCompr | User Subroutine | Specification** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **Compr** or **MCompr | User Subroutine | Specification** sheet. Aspen Plus does not retain these arrays from one call to the next.

23 User Solubility Subroutine for Crystallizer

You can supply a user solubility subroutine for the unit operation model Crystallizer.

The user solubility subroutine provides the saturation concentration to be used in determining crystal product flow rate in the model. The solubility subroutine can also be used to calculate crystal product flow rate, which replaces the model calculation.

To provide a user solubility subroutine for Crystallizer, select **User Subroutine** for **Saturation Calculation Method** on the **Crystallizer | Setup | Specifications** sheet and specify the subroutine name on the **Crystallizer | Advanced | User Subroutine** sheet.

Crystallizer Solubility Subroutine

Calling Sequence for Crystallizer Subroutine

```
SUBROUTINE subrname† (SOUTC, IPROD, ITYCON, NC, IDX, TEMP,
                        CONCEN, XMW, CSMIN, QLSOLN, FLSOLN,
                        QLSOLV, FLSOLV, FLSOLU, PRODIN, SLRYIN,
                        VENT, RHOL, RHOLL, FRHYD, NINTS, INTS,
                        NREALS, REALS, NIWKS, IWORKS, NWKS,
                        WORKS, CSAT, PROD, NSUB, IDXSUB, STOIC,
                        NTOT, SVENT, NCP, NCPNC, IDCRY)
```

[†] Subroutine name you entered on the **Crystallizer | Advanced | User Subroutine** sheet.

Argument List Descriptions for Crystallizer Subroutine

Variable	I/O [†]	Type	Dimension	Description and Range
SOUTC	I	REAL*8	—	Stream vector (see Appendix C)

Variable	I/O [†]	Type	Dimension	Description and Range
IPROD	O	INTEGER	—	Flag for calculating crystallizer yield 0 = calculated by model 1 = calculated in this routine
ITYCON	O	INTEGER	—	Type of concentration 0 = kg/m ³ of solution (or solvent) 1 = kg/kg of solution (or solvent)
NC	I	INTEGER	—	Total number of components present (conventional and non-conventional)
IDX	I	INTEGER	NCP	Component index vector (see IDX)
TEMP	I	REAL*8	—	Temperature (K)
CONCEN	I	REAL*8	NC	Concentrations (kg/m ³ or kg/kg, based on value of ITYCON)
XMW	I	REAL*8	NC	Molecular weight
CSMIN	I	REAL*8	—	Concentration of limiting component (kg/m ³ or kg/kg)
QLSOLN	I	REAL*8	—	Volume flow rate of solution (m ³ /s)
FLSOLN	I	REAL*8	—	Mass flow rate of solution (kg/s)
QLSOLV	I	REAL*8	—	Volume flow rate of solvent (m ³ /s)
FLSOLV	I	REAL*8	—	Mass flow rate of solvent (kg/s)
FLSOLU	I	REAL*8	—	Mass flow rate of solute (kg/s)
PRODIN	I	REAL*8	—	Mass flow rate of crystal entering Crystallizer (kg/s)
SLRYIN	I	REAL*8	—	Mass flow rate of slurry entering Crystallizer (kg/s)
VENT	I	REAL*8	—	Vent mass flow rate (kg/s)
RHOL	I	REAL*8	—	Density of solution (kg/m ³)
RHOLL	I	REAL*8	—	Density of solvent (kg/m ³)
FRHYD	I	REAL*8	—	Mass fraction of hydrated solvent
NINTS	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INTS	I	INTEGER	NINTS	Vector of integer parameters (see Integer and Real Parameters)
NREALS	I	INTEGER	—	Number of Real Parameters (see Integer and Real Parameters)
REALS	I	REAL*8	NREALS	Vector of Real Parameters (see Integer and Real Parameters)
NIWKS	I	INTEGER	—	Length of integer work vector (see Local Work Arrays)
IWORKS	I	INTEGER	NIWKS	Integer work vector (see Local Work Arrays)
NWKS	I	INTEGER	—	Length of real work vector (see Local Work Arrays)
WORKS	I	REAL*8	NWKS	Real work vector (see Local Work Arrays)
CSAT	O	REAL*8	—	Saturation concentration (kg/m ³ or kg/kg, based on value of ITYCON)
PROD	O	REAL*8	—	Crystal product flow rate (kg/s)
NSUB	I	INTEGER	—	Number of substreams
IDXSUB	I	INTEGER	NSUB	Substream index vector

Variable	I/O [†]	Type	Dimension	Description and Range
STOIC	I	REAL*8	NSUB, NC, 1	Stoichiometric coefficients for crystallization kinetics
NTOT	I	INTEGER	—	Length of a stream vector
SVENT	I	REAL*8	NTOT	Stream vector of vent flow
NCP	I	INTEGER	—	Number of components present in crystallization kinetics (conventional and non-conventional)
NCPNC	I	INTEGER	—	Number of non-conventional components present
IDCRY	I	INTEGER	—	Component index number (see IDX) of the crystal product

[†] I = Input to subroutine, O = Output from subroutine

IDX

IDX is a vector of length NCP containing component sequence numbers of the components present in the crystallization kinetics.

The values of IDX are component index numbers, where the components are numbered in the order they appear on the **Components | Specification | Selection** sheet, except that all conventional components are numbered first in the order shown, followed by non-conventional components present in the Crystallizer model, in the order shown.

For example, if NC = 6, and only the third and sixth components on the Components form are non-conventional (and present in the Crystallizer) and the rest are conventional components, then NCPNC = 2. The component numbers (in the order the components are listed on the Components form) are 1, 2, 5, 3, 4, 6. If the first, third, and fifth components on the Components form are involved in crystallization kinetics, then NCP = 3 and IDX = (1,4,5).

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **Crystallizer | Advanced | User Subroutine** sheet. You can initialize these parameters by assigning values on the same sheet. The default values are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **Crystallizer | Advanced | User Subroutine** sheet. Aspen Plus does not retain these arrays from one call to the next.

24 User Performance Curves Subroutine for Pump

This chapter describes the user performance curves subroutine for Pump.

To supply a user performance curves subroutine for Pump, select **User Subroutine** for **Curve Format** on the **Pump | Performance Curves | Curve Setup** sheet. On the **Pump | User Subroutine | Specification** sheet, choose the type of user curves and specify the user subroutine name.

The user subroutine calculates one of the following:

- Head.
- Head coefficient.
- Power.
- Discharge pressure.
- Pressure ratio.
- Pressure change.

In addition, you can also calculate efficiency and required net positive suction head (NPSHR) in the user subroutine.

Pump Performance Curve Subroutine

Calling Sequence for User Performance Curves Subroutine

```
SUBROUTINE subrname† (S1, NSUBS, IDXSUB, ITYPE, NBOPST, NINT,  
                        INT, NREAL, REAL, NIWORK, IWORK, NWORK,  
                        WORK, DIAMIM, VFLIN, ACTSPD, SUSPSP,  
                        IUSRCT, VALD1U, VALD2U, VALD3U)
```

[†] Subroutine name you entered on the **Pump | User Subroutine | Specification** sheet.

Argument List for User Performance Curves Subroutine

Variable	I/O [†]	Type	Dimension	Description and Range
S1	I	REAL*8	(1)	Stream vector after initial flash (has already accounted for pressure losses at suction nozzle) (see Appendix C)
NSUBS	I	INTEGER	—	Number of substreams
IDXSUB	I	INTEGER	NSUBS	Substream index vector
ITYPE	I	INTEGER	NSUBS	Substream type vector: 1,=MIXED, 2=CISOLID, 3=NC
NBOPST	I	INTEGER	6	Property option set
NINT	I	INTEGER	—	Number of integer parameters (see Integer and Real Parameters)
INT	I	INTEGER	NINT	Vector of integer parameters (see Integer and Real Parameters)
NREAL	I	INTEGER	—	Number of real parameters (see Integer and Real Parameters)
REAL	I	REAL*8	NREAL	Vector of real parameters (see Integer and Real Parameters)
NIWORK	I	INTEGER	—	Length of user integer work array (see Local Work Arrays)
IWORK	I	INTEGER	NIWORK	User integer work array (see Local Work Arrays)
NWORK	I	INTEGER	—	Length of user real work array (see Local Work Arrays)
WORK	I	REAL*8	NWORK	User real work area (see Local Work Arrays)
DIAMIM	I	REAL*8	—	Impeller diameter (m)
VFLIN	I	REAL *8	—	Volumetric flow rate at the suction condition (m ³ /s)
ACTSPD	I	REAL*8	—	Actual operating shaft speed (1/sec)
SUSPSP	I	REAL*8	—	Suction specific speed (units depend on speed units selected on Pump Setup Calculation Options sheet)
IUSRCT	I	INTEGER	—	Type of dependent variable (specified on the User Subroutine Specification sheet) 0=none 1=head 2=head coefficient 3=power 4=discharge pressure 5=pressure ratio 6=pressure change
VALD1U	O	REAL*8	—	Calculated value of the dependent variable (in SI units)
VALD2U	O	REAL*8	—	Calculated value of efficiency
VALD3U	O	REAL*8	—	Calculated value of NPSHR (m)

[†] I = Input to subroutine, O = Output from subroutine

Integer and Real Parameters

You can use integer and real retention parameters by specifying the number on the **Pump | User Subroutine | Specification** sheet. You can initialize these parameters by assigning values on the same sheet. The default values

are USER_RUMISS for real parameters and USER_IUMISS for integer parameters. Aspen Plus retains these parameters from one call to the next. The variables USER_IUMISS and USER_RUMISS are located in labeled common PPEXEC_USER (see Appendix A).

Local Work Arrays

You can use local work arrays by specifying the array length on the **Pump | User Subroutine | Specification** sheet. Aspen Plus does not retain these arrays from one call to the next.

25 User Subroutines for Petroleum Property Methods

Aspen Plus provides several correlations for basic properties, thermodynamic properties, and equation of state parameters for petroleum characterization. If these are not sufficient, you can provide your own correlation.

To provide your own correlation:

- 1 Write a Fortran subroutine to calculate the property or parameter using your correlations. The subroutine name must be the same as the name of the user routine you will select in step 2.
- 2 Go to the **Components | Petro Characterization | Property Methods | Basic** sheet. Specify a base property method. For the properties or parameters for which you want to supply your own correlation, select the User routine option.

The following pages list the calling arguments to the subroutines.

Calling Sequence for Boiling Point Subroutine

SUBROUTINE PCABPU (XMW, SG, ABP)

Argument List Descriptions for Boiling Point Subroutine

Variable	I/O [†]	Type	Dimension	Description
XMW	I	REAL*8	—	Molecular weight
SG	I	REAL*8	—	Specific gravity
ABP	O	REAL*8	—	Average boiling point

[†] I = Input to subroutine, O = Output from subroutine

Calling Sequence for Benedict-Webb-Rubin Parameter Subroutine

SUBROUTINE PCBWRU (ABP, API, SG, XMW, BWRGMA, TCBWR, VCBWR)

Argument List Descriptions for BWR Parameter Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
BWRGMA	O	REAL*8	—	BWR orientation parameter
TCBWR	O	REAL*8	—	BWR critical temperature
VCBWR	O	REAL*8	—	BWR critical volume

[†] I = Input to subroutine , O = Output from subroutine

Calling Sequence for Ideal Gas Heat Capacity Subroutine

SUBROUTINE PCCPGU (ABP, API, SG, XMW, TC, PC, CPIG)

Argument List Descriptions for Ideal Gas Heat Capacity Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
TC	I	REAL*8	—	Critical temperature
PC	I	REAL*8	—	Critical pressure
CPIG	O	REAL*8	11	Ideal gas heat capacity coefficients

[†] I = Input to subroutine, O = Output from subroutine

Calling Sequence for Gibbs Free Energy of Formation Subroutine

SUBROUTINE PCDGFU (ABP, API, SG, XMW, DGFORM)

Argument List Descriptions for Gibbs Free Energy of Formation Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
DGFORM	O	REAL*8	—	Gibbs free energy of formation

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Heat of Formation Subroutine

SUBROUTINE PCDHFU (ABP, API, SG, XMW, DHFORM)

Argument List Descriptions for Heat of Formation Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
DHFORM	O	REAL*8	—	Heat of formation

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Enthalpy of Vaporization Subroutine

SUBROUTINE PCDHVU (ABP, API, SG, XMW, TC, PC, DHVLB)

Argument List Descriptions for Enthalpy of Vaporization Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
TC	I	REAL*8	—	Critical temperature
PC	I	REAL*8	—	Critical pressure
DHVLB	O	REAL*8	—	Heat of vaporization

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Hydrocarbon Solubility Subroutine

SUBROUTINE PCHSLU (ABP, API, SG, XMW, HCSOL)

Argument List Descriptions for Viscosity Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
HCSOL	O	REAL*8	5	Hydrocarbon solubility parameters. The first three elements are A, B, and C in $\ln(x) = A + B/T + C \ln T$ and the last two elements are lower and upper temperature limits, as described in Hydrocarbon Solubility in online help.

[†] I = Input to subroutine, O = Output from subroutine

Calling Sequence for Viscosity Subroutine

SUBROUTINE PCMULU (ABP, API, SG, XMW, TC, TEMP, VISC)

Argument List Descriptions for Viscosity Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
TC	I	REAL*8	—	Critical temperature
TEMP	I	REAL*8	—	Temperature
VISC	O	REAL*8	—	Viscosity

[†] I = Input to subroutine, O = Output from subroutine

Calling Sequence for Molecular Weight Subroutine

SUBROUTINE PCMWU (ABP, SG, XMW)

Argument List Descriptions for Molecular Weight Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
SG	I	REAL*8	—	Specific gravity
XMW	O	REAL*8	—	Molecular weight

[†] I = Input to subroutine, O = Output from subroutine

Calling Sequence for Acentric Factor Subroutine

SUBROUTINE PCOMGU (TC, PC, PLXANT, ABP, SG, XMW, OMEGA)

Argument List Descriptions for Acentric Factor Subroutine

Variable	I/O [†]	Type	Dimension	Description
TC	I	REAL*8	—	Critical temperature
PC	I	REAL*8	—	Critical pressure
PLXANT	I	REAL*8	9	Extended Antoine vapor pressure parameters
ABP	I	REAL*8	—	Average boiling point
SG	I	REAL*8	—	Specific gravity
OMEGA	O	REAL*8	—	Acentric Factor

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Critical Pressure Subroutine

SUBROUTINE PCPCU (ABP, API, SG, XMW, PC)

Argument List Descriptions for Critical Pressure Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
PC	O	REAL*8	—	Critical pressure

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Vapor Pressure Subroutine

SUBROUTINE PCPLU (ABP, API, SG, XMW, TC, PC, VPT, VPRES)

Argument List Descriptions for Vapor Pressure Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
TC	I	REAL*8	—	Critical temperature
PC	I	REAL*8	—	Critical pressure
VPT	I	REAL*8	—	Vapor pressure temperature
VPRES	O	REAL*8	—	Vapor pressure

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for RKS Interaction Parameters Subroutine

```
SUBROUTINE PCRKIU (ABP, API, SG, XMW, DELTA, BPVAL, LH2S, LCO2,
                  LN2)
```

Argument List Descriptions for RKS Interaction Parameters Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
DELTA	I	REAL*8	—	Solubility parameter
BPVAL	O	REAL*8	3	RKS binary parameters
LH2S	I	LOGICAL	—	H2S present flag
LCO2	I	LOGICAL	—	CO2 present flag
LN2	I	LOGICAL	—	N2 present flag

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Specific Gravity Subroutine

```
SUBROUTINE PCSGU (ABP, API, SG, XMW)
```

Argument List Descriptions for Specific Gravity Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	O	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Critical Temperature Subroutine

SUBROUTINE PCTCU (ABP, API, SG, XMW, TC)

Argument List Descriptions for Critical Temperature Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
TC	O	REAL*8	—	Critical temperature

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Critical Volume Subroutine

SUBROUTINE PCVCU (ABP, SG, XMW, OMEGA, TC, PC, VC)

Argument List Descriptions for Critical Volume Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
OMEGA	I	REAL*8	—	Acentric factor
TC	I	REAL*8	—	Critical temperature
PC	I	REAL*8	—	Critical pressure
VC	O	REAL*8	—	Critical volume

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Liquid Molar Volume Subroutine

SUBROUTINE PCVOLU (ABP, API, SG, XMW, TC, PC, VC, ZC, TR, VOL)

Argument List Descriptions for Liquid Molar Volume Subroutine

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
TC	I	REAL*8	—	Critical temperature
PC	I	REAL*8	—	Critical pressure
VC	I	REAL*8	—	Critical volume
ZC	I	REAL*8	—	Critical compressibility
TR	I	REAL*8	—	Reduced temperature
VOL	O	REAL*8	—	Liquid molar volume

† I = Input to subroutine, O = Output from subroutine

Calling Sequence for Water Solubility Subroutines

SUBROUTINE PCWSLU (ABP, API, SG, XMW, TC, WSOLCC)

SUBROUTINE PCWSLU2 (ABP, API, SG, XMW, TC, WSOLCC, NWS, W, IDIM)

Argument List Descriptions for Water Solubility Subroutines

Variable	I/O [†]	Type	Dimension	Description
ABP	I	REAL*8	—	Average boiling point (K)
API	I	REAL*8	—	API gravity
SG	I	REAL*8	—	Specific gravity
XMW	I	REAL*8	—	Molecular weight
TC	I	REAL*8	—	Critical temperature (K)
WSOLCC	O	REAL*8	5	Water solubility correlation parameters
NWS	I	INTEGER	—	Number of user-entered water solubility data points
W	I	REAL*8	IDIM	Work space that can be used for intermediate calculation results, if needed
IDIM	I	INTEGER	—	Dimension for work space W. You should not write beyond the available work space.

† I = Input to subroutine, O = Output from subroutine

26 COM Unit Operation Interfaces

This chapter describes the Component Object Model (COM) interfaces used to develop new Aspen Plus custom unit operations. Aspen Plus supports new unit operation models developed using the following languages:

- Visual Basic
- C++
- C#
- J++

Aspen Plus supports version 1.0 of the CAPE-OPEN interface specification. Visit the CAPE-OPEN Laboratories Network web site www.Co-LaN.org for technical documentation describing the standards.

Users interested in writing CAPE-OPEN unit operation models should apply for membership in the CAPE-OPEN Laboratories Network. Individual membership is free and provides access to source code for example models.

The CAPE-OPEN interfaces allow the development of new unit operation models for use with any simulator that supports the CAPE-OPEN COM interface standard. The standard covers both Corba and COM technologies. Aspen Plus provides support for part of the COM specification.

There are two ways to create the model:

- Use an example model as a template.
- Start from scratch.

To write a new model from scratch create a new Visual Basic (or other language) project and then add a reference to the CAPE-OPEN Type Library corresponding to the version of the standard you want to use. You add the reference to the type library using the **References** command on the **Project** menu. This type library contains the definitions of all CAPE-OPEN interfaces defined in the selected version of the standard. In the rest of this chapter we will refer to the interfaces from version 1.0 of the standard.

Your model will run in both Sequential Modular and Equation Oriented modes within Aspen Plus, but in Equation-oriented mode it will run under the Aspen Plus perturbation layer. CAPE-OPEN interfaces for equation-oriented models are not supported.

Summary of the differences between 0.93 and 1.0 versions of the CAPE-OPEN standards that affect the implementation of CAPE-OPEN Unit Operations:

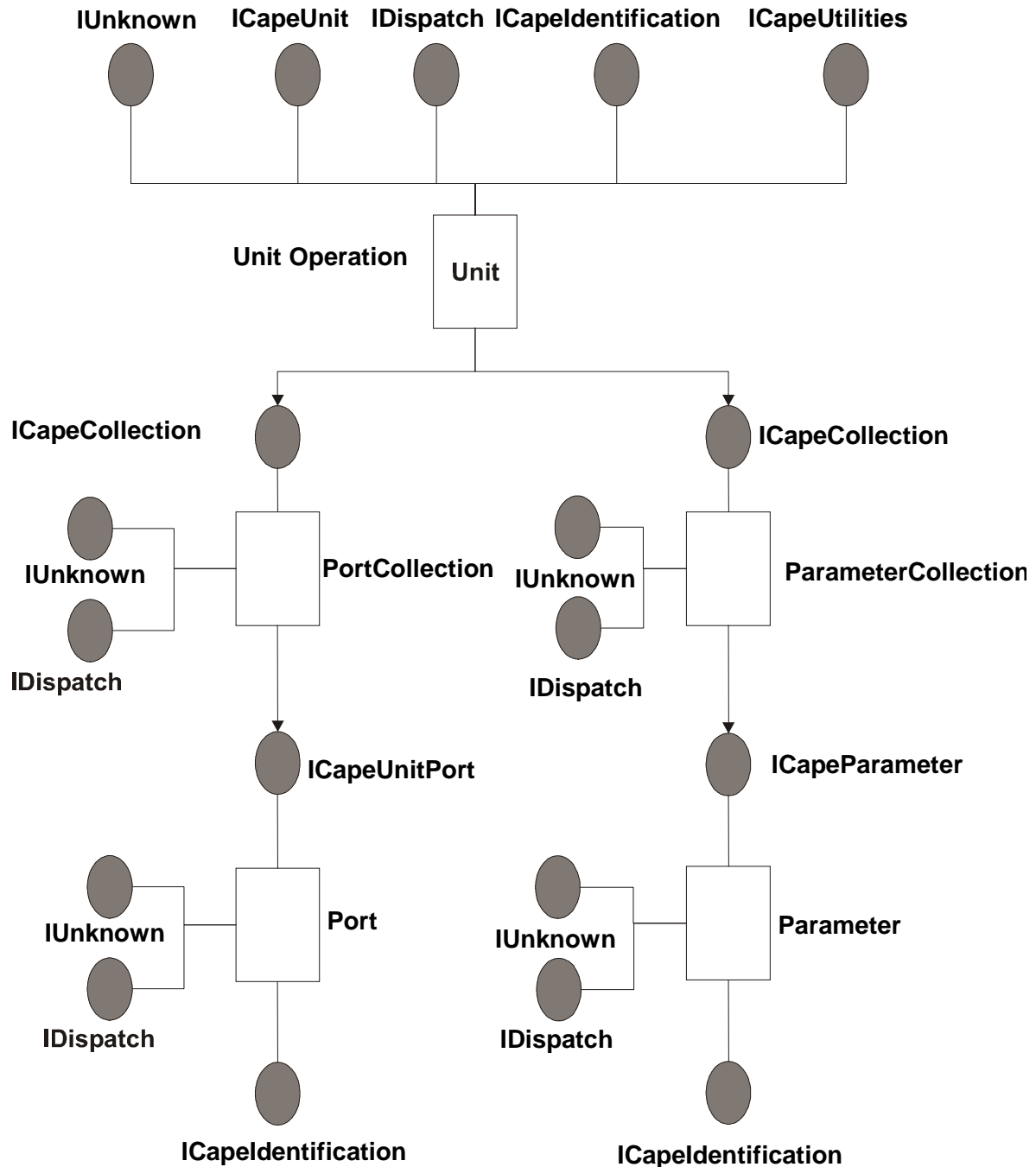
- CAPE-OPEN 1.0 introduces a new interface called ICapeUtilities. This is a general interface that may be implemented by any CAPE-OPEN Component. The Initialize, Edit, Terminate, SimulationContext and Parameters methods from the 0.93 version of the ICapeUnit interface have been moved to ICapeUtilities in version 1.0. As a consequence of this change the ICapeUnitEdit interface is removed altogether.
- The ICapeUnitCollection interface has been renamed ICapeCollection and is now a general interface intended for use with any type of collection.
- In the ECapeBadArgument interface the type of the "position" argument is changed from CapeLong to CapeShort for consistency with the written specification.
- The Simulation Context interfaces are defined. In particular the new ICapeDiagnostic interface allows a CAPE-OPEN component to generate output messages. Aspen Plus displays these messages in the Control Panel or the history file.

Components and Interfaces

The following table summarizes the components and associated CAPE-OPEN, Aspen Plus, and Microsoft COM interfaces required for Sequential Modular simulation:

Component	CAPE-OPEN Interfaces	Aspen Plus Interfaces	Microsoft Interfaces
Aspen Plus Simulator	ICapeDiagnostic	IAssayUpdate IATCapeXDiagnostic	
Unit(s)	ICapeUnit ICapeIdentification ICapeUtilities		IPersistStorage IPersistStream IPersistStreamInit IUnknown IDispatch
Port(s)	ICapeUnitPort ICapeIdentification		IUnknown IDispatch
Parameter(s)	ICapeParameter ICapeParameterSpec ICapeRealParameterSpec ICapeIntegerParameterSpec ICapeOptionParameterSpec ICapeIdentification	IATCapeXRealParameterSpec	IUnknown IDispatch
Collection(s)	ICapeCollection ICapeIdentification		IUnknown IDispatch

The following diagram displays the relationship between these components and the interfaces that each supports:



This chapter also describes the Aspen Plus interfaces that extend the CAPE-OPEN standard. CAPE-OPEN version 1.0 defines the **ICapeDiagnostic**, **ICapeCOSEUtilities**, and **ICapeMaterialTemplateSystem** interfaces through which a simulator can provide services to a CAPE-OPEN component. Aspen Plus provides the following interfaces which extend these services. Note that the **IATCapeXDiagnostic** interface functionality is now available through the

ICapeDiagnostic interface and that ICapeDiagnostic should be used in preference to IATCapeXDiagnostic. Support for IATCapeXDiagnostic is retained for backwards compatibility.

Interface	Description
IAssayUpdate	Provides access to simulation assay data. See Chapter 28.
IATCapeXRealParameterSpec	Allows the unit of measurement for a parameter to be defined. See page 296.
IATCapeXDiagnostic	Allows a CAPE-OPEN unit operation to write text to the Aspen Plus history file and control panel window. See page 295.

Note: Remember that these interfaces are specific to Aspen Plus. A COM model which relies on either the `IAssayUpdate` or the `IATCapeXDiagnostic` interface will only run in Aspen Plus.

The 1.0 version of the CAPE-OPEN standard requires that COM unit operations implement support for the standard `IPersist` mechanism used by other Microsoft COM components for persistence. Aspen Plus implements support for this mechanism and consequently, it expects a COM unit operation to implement one of the standard Microsoft persistence interfaces `IPersistStorage`, `IPersistStream`, or `IPersistStreamInit`. If a COM unit operation does not implement one of these interfaces, it will still run, but Aspen Plus assumes that it does not need to save its data.

Based on the programming language, the following is required:

Programming Language Required Action

Visual Basic	No action is required – the interfaces are automatically implemented.
C++	Write at least one of the interfaces according to Microsoft standards.

Aspen Plus includes a type library, `AspenCapeX`, which contains the definitions of all of the Aspen Plus-specific interfaces and enumerations described in this chapter. This type library is found in the `Engine\xeq` directory of the `APrSystem` installation. In addition, Aspen Plus installs the CAPE-OPEN type libraries for all three versions of the standard that are supported. These type libraries contain all the CAPE-OPEN interface and enumeration definitions. The libraries installed are:

```
C:\Program Files\Common Files\CAPE-OPEN\CAPE-OPENv1-0-0.tlb
C:\Program Files\Common Files\CAPE-OPEN\CAPE-OPENv0-9-3.tlb
C:\Program Files\Common Files\CAPE-OPEN\CAPE-OPENv0-9-0.tlb
```

Unit Interfaces

Unit interfaces include the following:

Interface Name	Description
ICapeUnit	Defines functions related to creating, deleting, and calculating a unit operation.
ICapeUtilities	Defines methods common to all CAPE-OPEN components – creation, deletion, editing and access to parameters
ICapeIdentification	Defines access functions for name and description. (See page 293.)

ICapeUnit Interface Methods

ICapeUnit consists of the following methods:

Method Name	Description
Calculate	Performs the unit operation model calculation.
ValStatus	Returns a flag indicating whether the unit is valid, invalid, or needs validating.
Ports	Returns an interface to a collection containing the unit's ports.
Validate	Returns a flag indicating whether the unit's data is valid.

Calculate

CAPE-OPEN Description

Calculate performs the unit operation model calculation, including progress monitoring and interruption checks (as required) using the simulation context.

Interface Name	ICapeUnit
Method Name	Calculate (no arguments)
Returns	CapeError

Implementation Details

For Aspen Plus, the `Calculate` method is expected to perform a flash calculation on the material objects associated with all the material outlet ports of the unit. See CAPE-OPEN COM Thermodynamic Interfaces, Chapter 27, for details of methods supported by the material object interfaces.

The Aspen Plus simulation context interfaces `IAssayUpdate` and `IATCapeXDiagnostic` do not allow progress monitoring or interrupt checking. A unit operation can write to the Model Manager Control Panel window to give information about progress if necessary, using methods from the `IATCapeXDiagnostic` interface.

If `Calculate` returns an error code:

- The CAPE-OPEN error interfaces associated with the error are queried for an explanation of the error which is written to the history file.
- The Aspen Plus outlet streams connected to the unit are not updated with data from the material objects associated with the unit's outlet ports.

ValStatus

CAPE-OPEN Description

`ValStatus` returns a flag of type `CapeValidationStatus` indicating whether the unit is valid, invalid, or needs validating. The possible values of the flag are:

- `CAPE_VALID`
- `CAPE_INVALID`
- `CAPE_NOT_VALIDATED`

Calling the `Validate` method is expected to set the unit's status to either `CAPE_VALID` or `CAPE_INVALID`, depending on whether the validation tests succeed or fail. Making a change to the unit operation, such as setting a parameter value, or connecting a stream to a port is expected to set the unit's status to `CAPE_NOT_VALIDATED`.

`Dirty` returns a flag indicating whether the unit has been modified since it was last saved. This flag is used to determine whether the unit's `Save` or `Validate` methods need to be called.

Interface Name	<code>ICapeUnit</code>
Method Name	<code>ValStatus</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
Status	0	CapeValidationStatus	CAPE_VALID if the unit is valid CAPE_INVALID if the unit has failed validation CAPE_NOT_VALIDATED if the unit has been updated since it was last validated.

Implementation Details

This method is not used by Aspen Plus. Persistence for COM models running in Aspen Plus is based on the standard `IPersistStorage` mechanism.

Ports

CAPE-OPEN Description

`Ports` returns an `ICapeCollection` interface that provides access to the unit's list of ports. Each element accessed through the returned interface must support the `ICapeUnitPort` interface.

Note: The `ICapeCollection` interface methods are described later in this chapter.

Interface Name	ICapeUnit
Method Name	Ports
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
PortsInterfaceO		CapeInterface	The interface of the collection containing the list of unit ports.

Validate

CAPE-OPEN Description

`Validate` returns a flag to indicate whether the unit is valid. This method should verify unit data, port data, and parameter values.

Interface Name	ICapeUnit
Method Name	Validate
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Message	O	CapeString	Message to explain the validation failure
IsOK	O	CapeBoolean	TRUE if the unit is valid.

ICapeUtilities Interface Method

ICapeUtilities consists of the following methods:

Method Name	Description
Initialize	Allows the unit to initialize itself.
Parameters	Returns an interface to a collection containing the unit's parameters.
SimulationContext	Assigns the simulation context to the unit.
Terminate	Allows the unit to release any allocated resources.
Edit	Displays the unit's user interface and allows the user to interact with it.

Initialize

CAPE-OPEN Description

`Initialize` allows the unit to initialize itself. It is called once when the unit operation model is instantiated in a particular flowsheet.

Note: Any initialization that could fail must be placed here instead of the constructor.

Interface Name	ICapeUtilities
Method Name	Initialize (no arguments)
Returns	CapeError

Implementation Details

The Initialize method must not change the current working directory: if it does Aspen Plus will not be able to access its own files and will fail. If the Initialize method needs to display a dialog to allow a user to open a file, it must ensure that the current working directory is restored after a file is selected.

Aspen Plus follows the `IPersist` protocol for constructing and initializing a COM unit operation.

- If the unit operation is a persistent Visual Basic class, Aspen Plus calls the `Class_InitProperties` method when the unit is instantiated.
- If the unit operation is a C++ class, which supports either the `IPersistStorage` or the `IPersistStreamInit` interface, Aspen Plus calls the `InitNew` method.

Parameters

CAPE-OPEN Description

`Parameters` returns an `ICapeCollection` interface that provides access to the unit's list of parameters. Each element accessed through the returned interface must support the `ICapeParameter` interface.

Note: The `ICapeCollection` interface methods are described later in this chapter.

Note: Aspen Plus expects that CAPE-OPEN unit operations do not have parameters whose names are the same except for capital/lower-case letters.

Interface Name	<code>ICapeUtilities</code>
Method Name	<code>Parameters</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
<code>PublicParams</code>	O	<code>CapeInterface</code>	The interface of the collection containing the list of unit parameters.

SimulationContext

CAPE-OPEN Description

`SimulationContext` assigns the simulation context to the unit.

Interface Name	<code>ICapeUtilities</code>
Method Name	<code>SimulationContext</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
<code>SimulationContext</code>	I	<code>CapeInterface</code>	The context of the simulator (progress, thermo, numeric).

Implementation Details

When Aspen Plus calls `SimulationContext` it passes an `IDispatch` interface to the unit. The unit can then query the `IDispatch` interface for the `ICapeDiagnostic`, `IAssayUpdate`, or `IATCapeXDiagnostic` interface.

`IAssayUpdate` provides access to Aspen Plus assay data. The `ICapeDiagnostic` and `IATCapeXDiagnostic` interfaces both allow messages to be written to the history file and control panel, and allows errors to be raised. The `ICapeDiagnostic` interface should be used in preference to the `IATCapeXDiagnostic` interface, which is supported for backwards compatibility only.

Aspen Plus does not implement support for the `ICapeMaterialTemplateSystem` and `ICapeCOSEUtilities` interfaces.

Terminate

CAPE-OPEN Description

`Terminate` releases any resources allocated by the unit. It verifies if the data has been saved and returns an error if not.

Interface Name	<code>ICapeUtilities</code>
Method Name	<code>Terminate</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
Message	I/O	<code>CapeString</code>	Message describing any error occurring during <code>Terminate</code> .
IsOK	O	<code>CapeBoolean</code>	Flag indicating whether termination succeeded or failed.

Implementation Details

Aspen Plus calls `Terminate` when a unit is deleted or a simulation is closed. The `Terminate` method is called immediately before one of the following:

- If the unit operation is a Visual Basic class, Aspen Plus calls the `Class_Terminate()` method when the unit is deleted or when a simulation is closed.
- If the unit operation is a C++ class, the class destructor is called when the unit is deleted or when the simulation is closed.

Edit

CAPE-OPEN Description

`Edit` displays the unit's user interface and allows the user to interact with it. If no user interface is available, it returns an error.

Interface Name	<code>ICapeUtilities</code>
----------------	-----------------------------

Method Name Edit
Returns CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Message	I/O	CapeString	Message describing any error occurring during edit.
IsOK	O	CapeBoolean	Flag indicating whether editing succeeded or failed.

Implementation Details

The Edit method must not change the current working directory: if it does Aspen Plus will not be able to access its own files and will fail. If the Edit method needs to display a dialog to allow a user to open a file, it must ensure that the current working directory is restored after a file is selected.

Port Interfaces

Port interfaces include the following:

Interface Name	Description
ICapeUnitPort	Defines functions to access the properties of a port.
ICapeIdentification	Defines access functions for name and description. (See page 0-293.)

ICapeUnitPort Interface Methods

ICapeUnitPort consists of the following methods:

Method Name	Description
Connect	Connects a stream to a port.
ConnectedObject	Returns the material, energy, or information object connected to the port using the <code>Connect</code> method.
Direction	Returns the direction in which objects or information connected to the port are expected to flow.
Disconnect	Disconnects the port from the connected stream.
PortType	Returns the type of the port.

Connect

CAPE-OPEN Description

`Connect` connects a stream to a port. The port validates the type of the object being passed.

Interface Name ICapeUnitPort

Method Name Connect
Returns CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
ObjectToConnect	I	CapeInterface	The object that is to be connected to the port.
Message	I/O	CapeString	Message describing any error occurring during connect.
IsOK	O	CapeBoolean	Flag indicating whether connection succeeded or failed.

Implementation Details

Aspen Plus creates a material object when it connects a stream to a port that belongs to a CAPE-OPEN unit. It then calls the port's `Connect` method passing in the material object's `IDispatch` interface. Aspen Plus gives the new material object the same name as stream that was connected to the port.

Material objects are described in CAPE-OPEN COM Thermodynamic Interfaces, Chapter 27.

ConnectedObject

CAPE-OPEN Description

`ConnectedObject` returns the material, energy, or information object connected to the port using the `Connect` method.

Interface Name ICapeUnitPort
Method Name ConnectedObject
Returns CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
ConnectedObject	O	CapeInterface	The object that is connected to the port.

Direction

CAPE-OPEN Description

`Direction` returns the direction in which objects or information connected to the port are expected to flow. The returned value must be one of the constants defined in the `CapePortDirection` enumeration, that is, one of `CapeInput`, `CapeOutput`, or `CapeInputOutput`.

Interface Name	ICapeUnitPort
Method Name	Direction
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
PortDirection	O	CapePortDirection	The direction of the port.

Disconnect

CAPE-OPEN Description

`Disconnect` disconnects the port from the connected stream.

Interface Name	ICapeUnitPort
Method Name	Disconnect
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Message	I/O	CapeString	Message describing any error occurring during disconnect.
IsOK	O	CapeBoolean	Flag indicating whether disconnection succeeded or failed.

Implementation Details

Aspen Plus calls `Disconnect` when a stream is disconnected from a port that belongs to a CAPE-OPEN unit.

PortType

CAPE-OPEN Description

`PortType` returns the type of the port. The returned value must be one of the constants defined in the `CapePortType` enumeration, that is, one of `CapeMaterial`, `CapeEnergy`, `CapeInformation`, or `CapeAny`.

Interface Name	ICapeUnitPort
Method Name	PortType
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
PortType	O	CapePortType	The type of the port.

Parameter Interfaces

Parameter interfaces include the following:

Interface Name	Description
ICapeParameter	Provides functions to access the properties of a parameter.
ICapeParameterSpec	Provides functions to access properties of a parameter specification that are independent of the type of the parameter value.
ICapeRealParameterSpec	Provides functions to access the properties of a specification for a real value.
ICapeIntegerParameterSpec	Provides functions to access the properties of a specification for an integer value.
ICapeOptionParameterSpec	Provides functions to access the properties of a specification for an option value. An option value is one of a set of fixed strings. For example, an option parameter might only allow the values "Yes" and "No".
ICapeIdentification	Defines access functions for name and description. (See page 0-293.)

ICapeParameter

`ICapeParameter` consists of the following methods:

Method Name	Description
ValStatus	Returns a flag indicating whether the parameter is valid, invalid, or needs validating.
Specification	Returns/assigns the specification of the parameter.
Validate	Checks whether the value of the parameter is valid in specification terms and in any other terms that are specified.
Value	Returns/assigns the value of the parameter.
ValueSource	Returns/assigns the source of the value for the parameter.

ValStatus

CAPE-OPEN Description

`ValStatus` returns a flag of type `CapeValidationStatus` indicating whether the parameter is valid, invalid, or needs validating. The possible values of the flag are:

- `CAPE_VALID`
- `CAPE_INVALID`

- CAPE_NOT_VALIDATED

Calling the Validate method is expected to set the parameter's status to either CAPE_VALID or CAPE_INVALID, depending on whether the validation tests succeed or fail. Making a change to the parameter value is expected to set the its status to CAPE_NOT_VALIDATED.

Interface Name	ICapeParameter
Method Name	ValStatus
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Status	O	CapeValidationStatus	CAPE_VALID if the unit is valid CAPE_INVALID if the unit has failed validation CAPE_NOT_VALIDATED if the unit has been updated since it was last validated.

Specification

CAPE-OPEN Description

`Specification` returns/assigns the specification of the parameter. The `[get]` method returns the specification as an interface to the correct specification type.

Note: If the specification type does not match, an error occurs (invalid argument).

Interface Name	ICapeParameter
Method Name	Specification
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Spec	I/O	CapeInterface	The specification interface of this parameter.

Validate

CAPE-OPEN Description

`Validate` ensures that the value of the parameter is valid in specification terms and in any other terms that are specified.

Interface Name	ICapeParameter
Method Name	Validate (no arguments)
Returns	CapeError

Value

CAPE-OPEN Description

Value returns/assigns the value of the parameter.

If this value is not passed as a `CapeVariant`, an error occurs (invalid argument).

Interface Name	ICapeParameter
Method Name	Value
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Value	I/O	CapeVariant	The value of the parameter.

ValueSource

CAPE-OPEN Description

ValueSource returns/assigns the source of the value for the parameter. This value can be:

- The user.
- An estimation.
- A calculated value.

Interface Name	ICapeParameter
Method Name	ValueSource
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
ValueSource	I/O	CapeLong	The source of the value of the parameter.

Implementation Details

This method is not used by Aspen Plus.

ICapeParameterSpec

ICapeParameterSpec consists of the following methods:

Method Name	Description
Type	Returns or assigns the type of the parameter specification.
Mode	Returns or assigns the access mode for the parameter.
Dimensionality	Returns or assigns the dimensionality for the parameter's value.

Type

CAPE-OPEN Description

Type returns or assigns the type of the parameter value. The parameter type can be one of CapeRealParam or CapeIntParam.

Interface Name	ICapeParameterSpec
Method Name	Type
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
ParamType	I/O	CapeParamType	The type of the parameter value.

Mode

CAPE-OPEN Description

Mode returns or assigns the access mode for a parameter value. The access mode can be one of CapeRead, CapeWrite, or CapeReadWrite.

Interface Name	ICapeParameterSpec
Method Name	Mode
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Mode	I/O	CapeParamMode	The access mode for the parameter value.

Implementation Details

The Aspen Plus data browser uses this method to fill in the Read Only field in the grid used to display parameter values. If `Mode` is set to `CapeRead` then the value for the parameter cannot be updated.

Dimensionality

CAPE-OPEN Description

Dimensionality returns or assigns the dimensionality of the parameter value. The dimensionality is represented as an array of values, one for each of the fundamental physical dimensions. The CAPE-OPEN standard leaves the exact specification of this representation open.

Interface Name	ICapeParameterSpec
Method Name	Dimensionality
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Dimensions I/O		CapeVariant	The dimensionality of the parameter value.

Implementation Details

Aspen Plus does not use this method. Instead a parameter can implement the `IATCapeXRealParameterSpec` interface which can be used to define the display unit for a parameter value.

ICapeRealParameterSpec Interface Methods

`ICapeRealParameterSpec` consists of the following methods:

Method Name	Description
DefaultValue	Returns/assigns the default value of the specified real valued parameter.
LowerBound	Returns/assigns the lower bound of the specified real valued parameter.
UpperBound	Returns/assigns the upper bound of the specified real valued parameter.
Validate	Validates a real value against its specification.

DefaultValue

CAPE-OPEN Description

`DefaultValue` returns/assigns the default value of the specified real valued parameter.

Interface Name	ICapeRealParameterSpec
Method Name	DefaultValue
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
DefaultValue	I/O	CapeDouble	The default value of the real valued parameter.

LowerBound

CAPE-OPEN Description

`LowerBound` returns/assigns the lower bound of the specified real valued parameter.

Interface Name	ICapeRealParameterSpec
Method Name	LowerBound
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
LBound	I/O	CapeDouble	The lower bound of the real valued parameter.

UpperBound

CAPE-OPEN Description

`UpperBound` returns/assigns the upper bound of the specified real valued parameter.

Interface Name	ICapeRealParameterSpec
Method Name	UpperBound
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
UBound	I/O	CapeDouble	The upper bound of the real valued parameter.

Validate

CAPE-OPEN Description

`Validate` validates a real value against its specification. It returns a flag indicating the success or failure of the validation together with a string that can be used to provide information about the validation.

Interface Name	ICapeRealParameterSpec
Method Name	Validate
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Value	I	CapeDouble	The value of the parameter to validate.
Message	O	CapeString	The message conveying information regarding the validation.
IsOK	O	CapeBoolean	A flag to indicate success or failure.

ICapeIntegerParameterSpec Interface Methods

`ICapeIntegerParameterSpec` consists of the following methods:

Method Name	Description
DefaultValue	Returns/assigns the default value of the specified integer valued parameter.
LowerBound	Returns/assigns the lower bound of the specified integer valued parameter.
UpperBound	Returns/assigns the upper bound of the specified integer valued parameter.
Validate	Validates an integer value against its specification.

DefaultValue

CAPE-OPEN Description

DefaultValue returns/assigns the default value of the specified integer valued parameter.

Interface Name	ICapeIntegerParameterSpec
Method Name	DefaultValue
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
DefaultValue	I/O	CapeLong	The default value of the integer valued parameter.

LowerBound

CAPE-OPEN Description

LowerBound returns/assigns the lower bound of the specified integer valued parameter.

Interface Name	ICapeIntegerParameterSpec
Method Name	LowerBound
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
LBound	I/O	CapeLong	The lower bound of the integer valued parameter.

UpperBound

CAPE-OPEN Description

UpperBound returns/assigns the upper bound of the specified integer valued parameter.

Interface Name	ICapeIntegerParameterSpec
Method Name	UpperBound
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
UBound	I/O	CapeLong	The upper bound of the integer valued parameter.

Validate

CAPE-OPEN Description

`Validate` validates an integer value against its specification. It returns a flag to indicate the success or failure of the validation together with string that can be used to provide information about the validation.

Interface Name	ICapeIntegerParameterSpec
Method Name	Validate
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Value	I	CapeLong	The value of the parameter being validated.
Message	O	CapeString	The message conveying information regarding the validation.
IsOK	O	CapeBoolean	Flag indicating whether validation succeeded or failed.

ICapeOptionParameterSpec Interface Methods

`ICapeOptionParameterSpec` consists of the following methods:

Method Name	Description
DefaultValue	Returns/assigns the default value of the specified option-valued parameter.
OptionList	Returns the valid options for this parameter as an array of strings.
RestrictedToList	Returns a boolean value to indicate whether values for the parameter are restricted to the specified options or not.
Validate	Validates an option value against its specification.

DefaultValue

CAPE-OPEN Description

`DefaultValue` returns/assigns the default value of the specified option-valued parameter.

Interface Name	ICapeOptionParameterSpec
Method Name	DefaultValue
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
DefaultValue	O	CapeString	The default value of the option-valued parameter.

OptionList

CAPE-OPEN Description

`OptionList` returns the list of valid options for the parameter as an array of strings.

Interface Name	ICapeOptionParameterSpec
Method Name	OptionList
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
OptionNames	O	CapeArrayString	The valid options for the parameter.

RestrictedToList

CAPE-OPEN Description

`RestrictedToList` returns a boolean value to indicate whether the value of the parameter must be one of the option values, or whether other values may be assigned.

Interface Name	ICapeOptionParameterSpec
Method Name	RestrictedToList
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Restricted	O	CapeBoolean	Returns TRUE if the parameter value must be one of the values returned by <code>OptionList</code> . Returns FALSE if other values are allowed.

Validate

CAPE-OPEN Description

`Validate` validates an option value against its specification. It returns a flag to indicate the success or failure of the validation together with string that can be used to provide information about the validation.

Interface Name	ICapeOptionParameterSpec
Method Name	Validate
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Value	I	CapeString	The value of the parameter being validated.
Message	O	CapeString	The message conveying information regarding the validation.
IsOK	O	CapeBoolean	Flag indicating whether validation succeeded or failed.

Collection Interfaces

Collection interfaces include the following:

Interface Name	Description
ICapeCollection	Provides functions to access the elements of a collection.
ICapeIdentification	Defines access functions for name and description. (See page 0-293.)

ICapeCollection Interface Methods

`ICapeCollection` consists of the following methods:

Method Name	Description
Count	Returns the number of objects in the collection.
Item	Returns an element from the port's collection or the collection of unit parameters.

Count

CAPE-OPEN Description

`Count` returns the number of objects in the collection.

Interface Name	ICapeCollection
Method Name	Count
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
ItemsCount	O	CapeLong	Number of items in the collection.

Item

CAPE-OPEN Description

`Item` returns an element from a collection. Pass either a name or position in the collection to identify the required element.

Interface Name	ICapeCollection
Method Name	Item
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Id	I	CapeVariant	Identifier for the requested item:
			Name of item
			Position in collection
Item	O	CapeInterface	The requested element.

ICapeIdentification Interface Methods

The `ICapeIdentification` interface is used for all the CAPE-OPEN COM components (including unit, port, parameter, and collection). Note that this interface does not belong to the UNIT System of interfaces, but to the overall CAPE-OPEN System.

`ICapeIdentification` consists of the following methods:

Method Name	Description
<code>ComponentDescription</code>	Returns/assigns the description of the component.
<code>ComponentName</code>	Returns/assigns the name of the component.

ComponentDescription

CAPE-OPEN Description

ComponentDescription returns/assigns the description of the component.

Interface Name	ICapeIdentification
Method Name	ComponentDescription
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Desc	I/O	CapeString	The description of the component.

ComponentName

CAPE-OPEN Description

ComponentName returns/assigns the name of the component.

Interface Name	ICapeIdentification
Method Name	ComponentName
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Name	I/O	CapeString	The name of the component.

Aspen Plus Interfaces

Aspen Plus interfaces include the following:

Interface Name	Description
IAssayUpdate	Provides access to simulation assay data. See COM Interface for Updating Oil Characterizations and Petroleum Properties, Chapter 28.
IATCapeXDiagnostic	Allows a CAPE-OPEN unit operation to write text to the Aspen Plus history file and control panel window.
IATCapeXRealParameterSpec	Defines the unit of measurement to use when displaying the value of a parameter in the Data Browser.

IATCapeXDiagnostic Interface Methods

Aspen Plus provides `IATCapeXDiagnostic` as an extension to the CAPE-OPEN standard. The interface is passed to the unit when Aspen Plus calls `SimulationContext`. Use this interface to write diagnostic messages to Aspen Plus.

`IATCapeXDiagnostic` consists of the following methods:

Method Name	Description
<code>SendMsgToHistory</code>	Writes text to the history file.
<code>SendMsgToTerminal</code>	Writes text to the control panel window.
<code>RaiseError</code>	Signals an error to Aspen Plus.

SendMsgToHistory

Description

`SendMsgToHistory` writes text to the Aspen Plus history file.

Interface Name	<code>IATCapeXDiagnostic</code>
Method Name	<code>SendMsgToHistory</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
Message	I	<code>CapeString</code>	The text being written to the history file.

SendMsgToTerminal

Description

`SendMsgToTerminal` writes text to the Aspen Plus control panel window.

Interface Name	<code>IATCapeXDiagnostic</code>
Method Name	<code>SendMsgToTerminal</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
Message	I	<code>CapeString</code>	The text being written to the terminal.

RaiseError

Description

`RaiseError` signals errors and warnings to Aspen Plus. Use this method if something goes wrong during the `Calculate` code.

Note: An error with severity `ErrorSeverityTerminal` terminates the Aspen Plus simulation.

Interface Name	IATCapeXDiagnostic
Method Name	RaiseError
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
Severity	I	ErrorSeverity, one of: ErrorSeverityTerminal ErrorSeveritySevere ErrorSeverityError ErrorSeverityWarning	The severity of the error.
Context	I	CapeString	A string which identifies the unit.
Message	I	CapeString	The text being written to the history file.

IATCapeXRealParameterSpec

`IATCapeXRealParameterSpec` consists of the following method:

Method Name	Description
DisplayUnits	Defines the display unit for the parameter.

DisplayUnits

Description

`DisplayUnits` defines the unit of measurement symbol for a parameter.

Note: The symbol must be one of the uppercase strings recognized by Aspen Plus to ensure that it can perform unit of measurement conversions on the parameter value. The system converts the parameter's value from SI units for display in the data browser and converts updated values back into SI.

Interface Name	IATCapeXRealParameterSpec
Method Name	DisplayUnits
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
UOMString	O	CapeString	A string of uppercase characters containing the unit of measurement symbol used by Aspen Plus to display the value of the parameter.

Implementation Details

Aspen Plus stores all values in SI units internally. It converts values from SI units to display units when they are displayed, and it converts values from display units to SI units when they are changed. Use

`IATCapeXRealParameterSpec` to perform the same conversions on the unit operation's parameters.

Using this interface means that the upper bound, lower bound and value of a parameter are all in SI units. This makes these values consistent with the values returned from material objects. The exception is for mole-based units where Aspen Plus uses kmol for consistency with the standard mass unit kg, but the standard SI unit is the mole and CAPE-OPEN uses moles. Mole-based values must be converted appropriately when going between Aspen Plus and CAPE-OPEN data.

The following table lists the valid units of measurement symbols:

Physical Type	Valid Units of Measurement
ANGLE	Rad, Deg
AREA	Sqm, sqft
AREA-PRICE	\$/sqm, \$/sqft
AREA-USAGE	sqm/sec, sqft/hr, sqm/hr
BOND-WORK-IN	j/kg, kwhr/ton
CHROM-VEL	m/sec, ft/sec, cm/hr
COMPOSITION	mol-fr
CONTENTS	fraction, percent
CURRENT	amp, mamp
DENSITY	kg/cum, lb/cuft, gm/cc
DIFFUSIVITY	sqm/sec, sqft/hr, sqcm/sec
DIMENSIONLES	unitless
DIPOLEMOMENT	(j*cum)**.5, (btu*cuft)**.5, debye
ELEC-POWER	watt, kw
ELEC-PRICE	\$/j, \$/kwhr
ENERGY	j, btu, cal
ENERGY-PRICE	\$/j, \$/btu, \$/cal
ENTHALPY	j/kmol, btu/lbmol, cal/mol
ENTHALPY-CYC	watt/cycle, btu/cycle, cal/cycle
ENTHALPY-FLO	watt, btu/hr, cal/sec
ENTHALPY-OPR	watt/cycle, btu/op-hr, cal/op-sec
ENTROPY	j/kmol-k, btu/lbmol-r, cal/mol-k
F-FACTOR	(kg-cum)**.5/se, (lb-cuft)**.5/hr, (gm-l)**.5/min

Physical Type	Valid Units of Measurement
FILTER-RESIS	1/meter, 1/ft
FISCAL	\$
FLOW	kg/sec, lb/hr, kg./hr
FLUX	cum/sqm-sec, cuft/sqft-sec, l/sqm-sec
FORCE	newton, lbf, dyne
FREQUENCY	hz, rpm
HEAD	j/kg, ft-lbf/lb, m-kgf/kg
HEAT	j, btu, cal
HEAT-FLUX	watt/m, btu/hr-ft, cal/sec-m
HEAT-TRANS-C	watt/sqm-k, btu/hr-sqft-r, cal/sec-sqcm-k
INVERSE-AREA	1/sqm, 1/sqft
INVERSE-HT-C	sqm-k/watt, hr-sqft-r/btu, sec-sqcm-k/cal
INVERSE-LENG	1/m, 1/ft, 1/cm
INVERSE-PRES	sqm/n, 1/psi, 1/atm
INVERSE-TEMP	1/k, 1/r
INVERSE-TIME	1/sec, 1/hr
ITEM-PRICE	\$/item
LENGTH	meter, ft
LN-INV-TIME	ln(1/sec), ln(1/hr)
MASS	kg, lb
MASS-CONC	kg/cum, lb/cuft, gm/l
MASS-CYCL	kg/cycle, lb/cycle
MASS-DENSITY	kg/cum, lb/cuft, gm/cc
MASS-ENTHALP	j/kg, btu/lb, cal/gm
MASS-ENTROPY	j/kg-k, btu/lb-r, cal/gm-k
MASS-FLOW	kg/sec, lb/hr, kg/hr
MASS-FLUX	kg/sqm-s, lb/sqft-hr, kg/sqm-hr
MASS-HEAT-CA	j/kg-k, btu/lb-r, cal/gm-k
MASS-OPER	kg/op-sec, lb/op-hr, kg/op-hr
MASS-PER-LEN	kg/m, lb/ft, kg/m
MASS-TRANS-C	kg/s-swm-kg/c, lb/hr-sqf-lb/c, gm/s-sqcm-gm/cc
MASS-VOLUME	cum/kg, cuft/lb, cc/kg
MOL-FLOW-LEN	kmol/sec-m, lbmol/hr-ft, kmol/hr-m
MOLE-CONC	kmol/cum, lbmol/cuft, mol/cc
MOLE-CYCL	kmol/cycle, lbmol/cycle
MOLE-DENSITY	kmol/cum, lbmol/cuft, mol/cc
MOLE-ENTHALP	j/kmol, btu/lbmol, cal/mol
MOLE-ENTROPY	j/kmol-k, btu/lbmol-r, cal/mol-k
MOLE-FLOW	kmol/sec, lbmol/hr, kmol/hr
MOLE-HEAT-CA	j/kmol-k, btu/lbmol-r, cal/mol-k
MOLE-OPER	kmol/op-sec, lbmol/op-hr, kmol/op-hr
MOLE-VOLUME	cum/kmol, cuft/lbmol, cc/mol
MOLES	kmol, lbmol

Physical Type	Valid Units of Measurement
MOM-INERTIA	kg/sqm, lb-sqft
NUM-CON-RATE	no/cum-sec, no/cuft-sec, no/l-sec
NUM-CONC	no/cum, no/cuft, no/l
PACK-FACTOR	1/m, 1/ft
PDROP	n/aqm, psi, atm
PDROP-PER-HT	n/cum, in-water/ft, mm-water/m
POP-DENSITY	no/m/cum, no/ft/cuft, no/m/l
POWER	watt, hp, kw
POWER-VOLUME	watt/cum, hp/cuft, kw/l
PRESSURE	n/sqm, psi, atm
RHO-VSQRD	kg/m-sqsec, lb/ft-sqsec, kg/m-sqsec
SOLUPARAM	(j/cum)**.5, (BTU/Cuft)**.5, (Cal/cc)**.5
SOLUTE-PERM	sqm/m-s, sqft/ft-hr, sqm/m-hr
SOLVENT-PERM	kg/sqm-s-pa, lb/sqft-hr-atm, KG/SQM-HR-ATM
SOUND-LEVEL	DECIBELS
SPEC-FLT-RES	meter/kg, ft/lb, meter/kg
SPECIFICAREA	sqm/cum, sqft/cuft, sqcm/cc
SURFACE-TENS	n/m, dyne/cm
TEMP-VOLUME	CUM-K/KMOL, CUFT-R/LBMOL, CC-K/MOL
TEMPERATURE	k, f
THERMAL-COND	watt/m-k, btu-ft/hr-sqft, kcal-m/hr-sqm-k
TIME	sec, hr
UA	j/sec-k, btu/hr-r, cal/sec-k
UNIT-PRICE	\$/kg, \$/lb
VELOCITY	m/sec, ft/sec
VFLOW-LENGTH	sqm/sec, gpm/ft, sqcm/sec
VFLOW-RPM	cum/sec/rpm, cuft/hr/rpm, l/min/rpm
VISCOSITY	n-sec/sqm, cp
VOL-ENTHALPY	j/cum, btu/cuft, cal/cc
VOL-HEAT-CAP	j/cum-k, btu/cuft-r, cal/cc-k
VOLTAGE	volt, kvolt
VOLUME	cum, cuft, l
VOLUME-CYCL	cum/cycle, cuft/cycle, l/cycle
VOLUME-FLOW	cum/sec, cuft/hr, l/min
VOLUME-OPER	cum/op-sec, cuft/op-hr, l/op-min
VOLUME-PRICE	\$/cum, \$/cuft, \$/l
VOLUME-USAGE	cum/sec, cuft/hr, l/hr
WATER-RATE	kg/j, lb/hp-hr, kg/kw-hr
WORK	j, hp-hr, kw-hr

Installation of COM Unit Operations

Microsoft COM supports component categories so that applications can easily identify classes that implement specific interfaces. Aspen Plus identifies CAPE-OPEN compliant COM unit operations by searching the registry for classes in the CAPE-OPEN unit operation category. The identifier (CATID) for this category is:

678c09a5-7d66-11d2-a67d-00105a42887f

Note: The Aspen Plus installation will automatically register the Category ids required by CAPE-OPEN if it is necessary.

The mechanism for registering a COM unit operation in the CAPE-OPEN unit operation category depends on whether the unit is implemented in C++ or Visual Basic.

For a C++ class:

Write the `DLLRegisterServer` method so that it adds the CAPE-OPEN unit operation CATID to the list of implemented categories for the class. Use the `regsvr32` utility to register the DLL built by the C++ compiler.

For a Visual Basic class:

If you use the Visual Basic Unit Operation Wizard it will create an install package for your Unit Operation that will create the correct registry entries. In this case no further action is required.

If you do not use the Wizard then Visual Basic will register the class automatically when it builds the DLL, but it does not add the CAPE-OPEN unit operation category to the registry. This entry has to be added separately. There are two ways to do this:

- Use the `regsvr32` utility to register the DLL built by Visual Basic.
- Create a `.reg` file containing the necessary entries.

Creating a .reg file

Copy `CapeDescription.reg` from your `C:\Program Files\Common Files\AspenTech Shared\CAPE-OPEN Example Models` directory (where C: represents the Windows installation drive) to the directory containing your Visual Basic project. Then edit this file and make changes appropriate for your Unit Operation. Finally, build your Visual Basic project and then double-click on your copy of the `.reg` file in Windows Explorer to add the entries to the registry.

Distributing COM Models to Users

If you are using the Unit Operation Wizard you can give the install package that it creates to other users so that they can install the Unit Operation on other machines. The Wizard provides documentation describing how to do this.

To distribute and install a unit operation that doesn't have an installation package onto another user's machine, register the DLL file using the regsvr32 utility described previously.

- 1 Copy the DLL to a directory on the target machine, and log into the machine.
- 2 Run the regsvr32 utility on the DLL.


Adding Compiled COM Models to the Aspen Plus Model Palette

Use a COM unit operation in Aspen Plus by selecting it from the CAPE-OPEN tab within the Model Palette and dragging it into the Process Flowsheet window.

To display the CAPE-OPEN tab in the Model Palette:

- 1 Start Aspen Plus.
- 2 From the **Developer** tab of the ribbon, click **Manage Libraries**.
- 3 In the **Manage Libraries** dialog box, check the **Is In Use** box next to **CAPE-OPEN** and click **OK**.

Aspen Plus displays a new tab called CAPE-OPEN on the Model Palette along with the other the built-in Model Palette tabs.

Note: If the tab is not visible, click  at the bottom of the screen until the tab appears.

Important: To define icons for COM models or to categorize them using different tabs, first create a user model library. Once the new library is created, copy COM models into the new library and modify the icons and tabs as necessary. See Chapter 16 of the *Aspen Plus User Guide* for more information.

Version Compatibility for Visual Basic COM Models

Every COM unit operation has a unique class identifier (CLSID). Aspen Plus stores this identifier in .appdf, .bkp, and .inp files as part of persisting a COM unit operation. For Visual Basic COM unit operations, the Visual Basic compiler assigns this identifier and changes it with each build of the unit operation. For C++ classes, the class identifier only changes when the model developer changes it.

The effect of changing the class identifier is that Aspen Plus cannot load files containing the previous class identifier. Text files can be corrected using a text editor but .appdf binary files cannot be corrected.

To prevent this problem, Visual Basic 6 uses the Version Compatibility option to fix the class identifier for a class.

Important: Always set this option to Project Compatibility for COM unit operations. This allows Visual Basic to use the class identifier from the previous version of the DLL built by the project.

To set this option in Visual Basic:

- 1 From the **Project** menu, select **Properties**.
- 2 From the **Project Properties** dialog, select the **Component** tab.
- 3 Select **Project Compatibility** from the list of compatibility options.

Note: This option is only available after the DLL file has been built for the first time.

Uninstalling COM Models

If your Unit Operation has an installation package use **Add/Remove Programs** from the Windows Control Panel to remove it from a machine.

Remove a COM unit operation that doesn't have an installation package from your machine by deleting its registry entries using the `regsvr32` utility from Microsoft.

- 1 Open a command prompt window.
- 2 Change the working directory to the directory where the DLL is located.

For example: `cd \My Projects\Membrane`

- 3 Execute the command:

```
Regsvr32 VBMembraneDLL /u
```

This command removes the entries associated with all the classes in VBMembraneDLL from the registry.

Note: Once a COM unit operation is uninstalled, Aspen Plus cannot load associated simulations.

27 CAPE-OPEN COM Thermodynamic Interfaces

This chapter describes CAPE-OPEN COM interfaces and the methods available for thermophysical properties and phase equilibrium calculations. These interfaces follow the standards defined in the Physical Properties interface Model and Specification document defined by CAPE-OPEN.

Aspen Plus now supports version 1.0 of the CAPE-OPEN standard. This version is very similar to the version 0.93 supported in Aspen Plus 11.1.

The benefits of using these open standard Thermodynamic interfaces are

- Unit operation model can use any physical property systems that are CAPE-OPEN compliant, without having to customize the model for each property system.
- Property model developer needs to develop the model using the interfaces only once. The model can then be used with any physical property system that is CAPE-OPEN compliant, without having to customize the model for each property system. This allows property specialists in the academia or industry to focus on model development work rather than on writing interfaces.

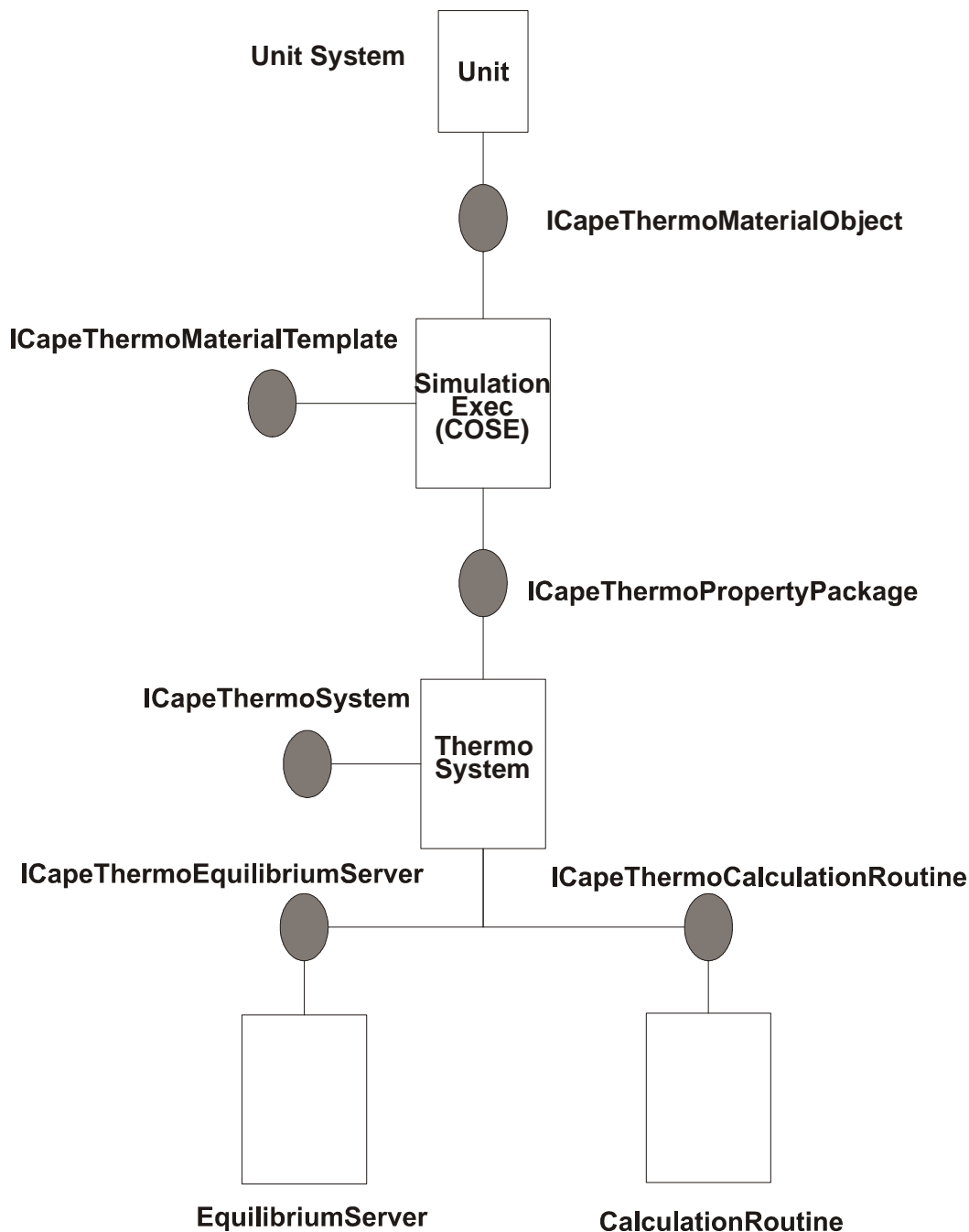
The following table summarizes the components associated with the CAPE-OPEN COM Thermodynamic interfaces:

Interface	Associated Component
ICapeThermoMaterialTemplate	Material Template
ICapeThermoMaterialObject	Material Object
ICapeThermoSystem	Physical Property System
ICapeThermoPropertyPackage	Property Package
ICapeThermoCalculationRoutine	Thermophysical Property Calculation Routines
ICapeThermoEquilibriumServer	Equilibrium Server (for flash calculations)

All CAPE-OPEN Thermodynamic interface methods are invoked via the material object, which is an instance of the material template. The physical property system owns the property package, which is responsible for the actual calculations of thermophysical properties and phase equilibrium. All calculations are usually performed by the native property system, however

CAPE-OPEN allows the use of external property calculation routines and equilibrium server in a particular property package.

The following diagram displays the relationship between the CAPE-OPEN Simulation Executive (COSE), the unit operation models, and the various components previously defined.



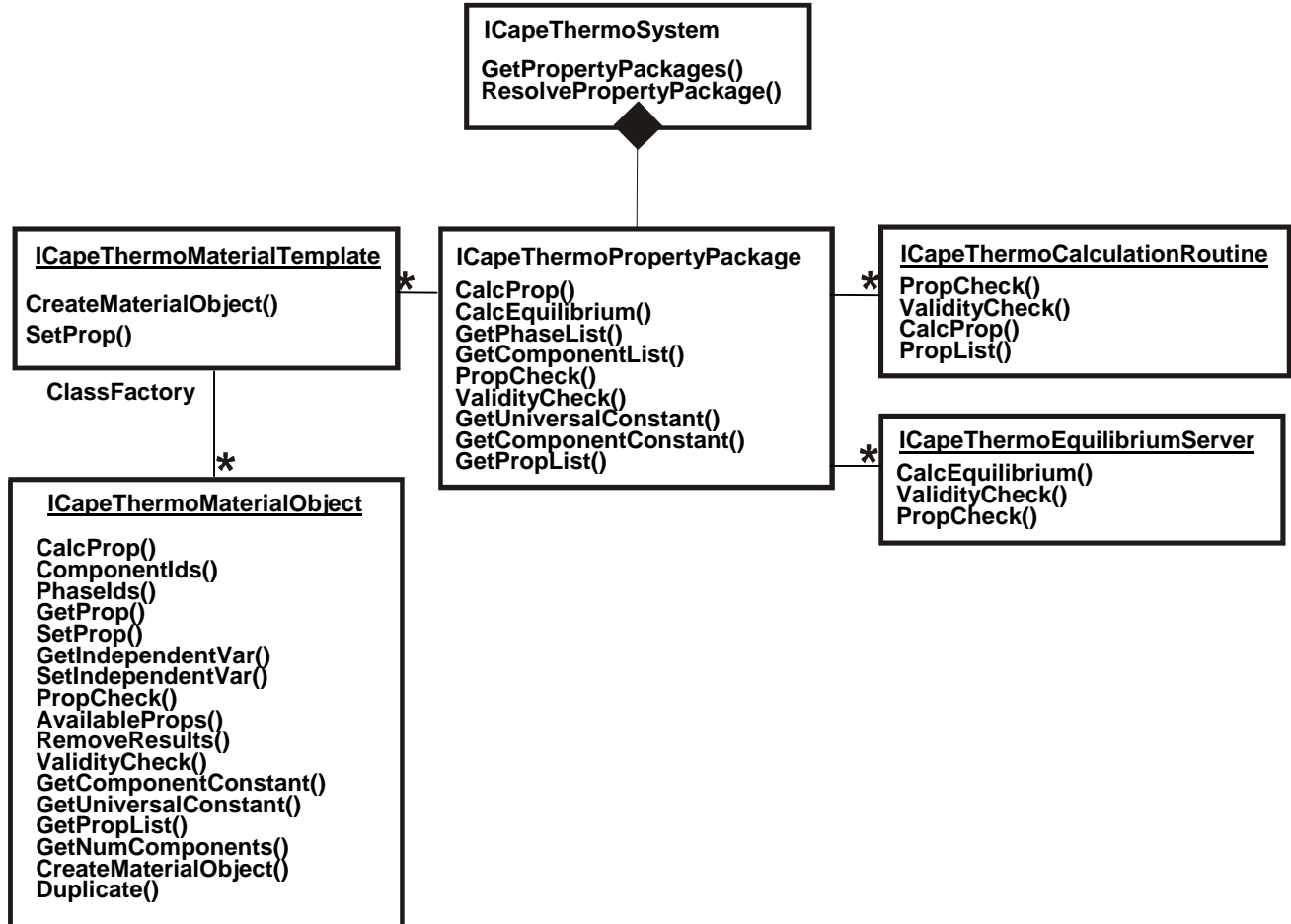
This diagram displays the interfaces supported by each of the components. The associations, represented by the lines from the components to the interfaces, are also detailed. The way these associations are implemented is

not dictated by CAPE-OPEN, but is proprietary to the component/simulation vendor.

The CAPE-OPEN compliant Simulation Executive (COSE):

- Maintains interface implementations of the material template and material object.
- Provides functionality for defining the material template.
- Delegates the material object to the appropriate Thermodynamic System and property package interfaces.

The following diagram summarizes CAPE-OPEN Thermodynamic interfaces:



Material Templates

Material templates define the characterization of a material. A material template is a set of services, usually provided by the COSE, required for the property package to work in coordination with unit operations and other pieces of the simulation.

Material templates provide unified interfaces for the following functions:

- 1 Association between a material object and a property package.** The active material template within the flowsheet provides the connection mechanism between any material object and the active property package within that part of the flowsheet.
- 2 Delegation of functions.** The material template handles the delegation of interface methods from the material object to the property package. The delegation uses the association between a material object and a property package. For example, in a PH flash calculation, the property package itself implements the code, but the material object interface provides the `CalcEquilibrium` method, which is then delegated to the property package.
- 3 Component mapping.** Different property packages active in different sections of the flowsheet may use different naming conventions for components. The material template implements the required component mapping.
- 4 Material object factory.** The material template implements the mechanism to create a material object.
- 5 Component subsetting.** The material template provides mechanisms used to implement further configuration of a property package to either reduce the number of components, or to modify the number of active phases.

The material template definition consists of:

- o A Component List
- o A Phase Assumption
- o Reference to CAPE-OPEN property package
- o List of custom attributes for pseudo-components

Note: The implementation of Material Templates is internal to Aspen Plus and is not directly accessible to CAPE-OPEN components.

Material Objects

Material objects define an instance of a material and are created from material templates.

A material object is an instance of the material template and is a CAPE-OPEN standard view of the proprietary streams present in Aspen Plus. The material object is a persistent entity that provides the data source necessary for the property package to perform its calculations, and provides a place to store results.

A material object is associated with each connected port on a CAPE-OPEN Unit Operation to store data characterizing the stream conditions at each port and to provide the unit operation with a way to request physical property and flash calculations.

The property package uses data specified in the material object (e.g., temperature, pressure, and component molar flow rates) to calculate the required properties.

ICapeThermoMaterialObject Interface Methods

The ICapeThermoMaterialObject interface consists of the following methods:

Method Name	Description
AvailableProps	Gets a list of calculated properties.
CalcEquilibrium	Delegates flash calculations to the associated Thermodynamic System.
CalcProp	Performs all property calculations and delegates the calculations to the associated Thermodynamic System.
CreateMaterialObject	Creates a material object from the parent material template of the current material object.
Duplicate	Creates a duplicate of the current material object.
GetComponentConstant	Retrieves component constants from the property package.
GetIndependentVar	Returns the independent variables of a material object.
GetNumComponents	Returns the number of components in a material object.
GetProp	Retrieves the calculation results from the material object.
GetPropList	Returns a list of properties supported by the property package and corresponding CO calculation routines.
GetUniversalConstant	Retrieves universal constants from the property package.
ICapeThermoMaterialObject	Returns the list of components Ids of a given material object.
PhaseIds	Returns the list of phases supported by the material object.
PropCheck	Verifies that given properties can be calculated.
RemoveResults	Removes all or specified property results in the material object.
SetIndependentVar	Sets the independent variable for a given material object.
SetProp	Sets the property values of the material object.
ValidityCheck	Checks the validity of the calculation.

AvailableProps

Description

`AvailableProps` gets a list of calculated properties.

Interface Name	ICapeThermoMaterialObject
Method Name	AvailableProps
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*props	O	CapeArrayString	Properties for which results are available.

CalcEquilibrium

Description

`CalcEquilibrium` delegates flash calculations to the associated Thermodynamic System.

Interface Name	ICapeThermoMaterialObject
Method Name	CalcEquilibrium
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
flashType	I	CapeString	Flash calculation type.
props	I	CapeVariant (String Array)	Properties to be calculated at equilibrium. NULL for no properties.

CalcProp

Description

`CalcProp` performs all property calculations and delegates the calculations to the associated Thermodynamic System.

Interface Name	ICapeThermoMaterialObject
Method Name	CalcProp
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
props	I	CapeVariant (String Array)	The List of Properties to be calculated.
phases	I	CapeVariant (String Array)	List of phases for which the properties are to be calculated.
calcType	I	CapeString	Type of calculation: Mixture Property or Pure Component Property. For partial property, such as fugacity coefficients of components in a mixture, use "Mixture" CalcType. For pure component fugacity coefficients, use "Pure" CalcType.

CreateMaterialObject

Description

CreateMaterialObject creates a material object from the parent material template of the current material object.

Interface Name	ICapeThermoMaterialObject
Method Name	CreateMaterialObject
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
materialObject	O	*ICapeInterface	The created/initialized material object.

Duplicate

Description

Duplicate creates a duplicate of the current material object.

Interface Name	ICapeThermoMaterialObject
Method Name	Duplicate
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
clone	O	*ICapeInterface	The created/initialized material object.

GetComponentConstant

Description

`GetComponentConstant` retrieves component constants from the property package.

Interface Name	ICapeThermoMaterialObject
Method Name	GetComponentConstant
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
props	I	CapeVariant (String Array)	List of component constants.
compIds	I	CapeVariant (String Array)	List of component IDs for which constants are to be retrieved. NULL for all components in the material object.
*propvals	O	CapeArrayDouble	Component Constant values returned from the property package for all the components in the material object.

GetIndependentVar

Description

`GetIndependentVar` returns the independent variables of a material object.

Interface Name	ICapeThermoMaterialObject
Method Name	GetIndependentVar
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
indVars	I	CapeVariant (String Array)	Independent variables to be set (see names for state variables for list of valid variables).
*values	O	CapeArrayDouble	Values of independent variables.

GetNumComponents

Description

`GetNumComponents` returns the number of components in a material object.

Interface Name	ICapeThermoMaterialObject
Method Name	GetNumComponents
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*num	O	CapeLong	Number of components in the material object.

GetProp

Description

`GetProp` retrieves the calculation results from the material object.

Interface Name	ICapeThermoMaterialObject
Method Name	GetProp
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
property	I	CapeString	The Property for which results are requested from the material object.
phase	I	CapeString	The qualified phase for the results.
compIds	I	CapeVariant (String Array)	The qualified components for the results. NULL to specify all components in the material object. For mixture property such as liquid enthalpy, this qualifier is not required. Use NULL as placeholder.
calcType	I	CapeString	The qualified type of calculation for the results. Valid calculation types include: Pure Mixture
basis	I	CapeString	Qualifies the basis of the result (i.e., mass /mole). Default is mole. Use NULL for default or as place holder for property for which basis does not apply.
*results	O	CapeArrayDouble	Results vector containing property values in SI units arranged by the defined qualifiers.

Note: `GetProp` returns mole-based quantities in mole units rather than kmol as Aspen Plus does. Mole is the standard SI unit but Aspen Plus uses kmol for consistency with the standard mass unit kg.

GetPropList

Description

`GetPropList` returns a list of properties supported by the property package and corresponding CO calculation routines.

Interface Name	ICapeThermoMaterialObject
Method Name	GetPropList
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*props	O	CapeArrayString	String list of all supported properties of the property package.

GetUniversalConstant

Description

`GetUniversalConstant` retrieves universal constants from the property package.

Interface Name	ICapeThermoMaterialObject
Method Name	GetUniversalConstant
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
props	I	CapeVariant (String Array)	List of universal constants to be retrieved.
*propvals	O	CapeArrayDouble	Values of universal constants.

ICapeThermoMaterialObject

Description

`ICapeThermoMaterialObject` returns the list of components Ids of a given material object.

Interface Name	ICapeThermoMaterialObject
Method Name	ICapeThermoMaterialObject
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*compIds	O	CapeVariant (String Array)	Component IDs.

PhaseIds

Description

PhaseIds returns the list of phases supported by the material object.

Interface Name	ICapeThermoMaterialObject
Method Name	PhaseIds
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*phaseIds	O	CapeVariant (String Array)	List of phases

PropCheck

Description

PropCheck verifies that given properties can be calculated.

Interface Name	ICapeThermoMaterialObject
Method Name	PropCheck
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
props	I	CapeVariant (String Array)	Properties to check.
*valid	O	CapeArrayBoolean	Returns Boolean List associated to list of properties to be checked.

RemoveResults

Description

`RemoveResults` removes all or specified property results in the material object.

Interface Name	ICapeThermoMaterialObject
Method Name	RemoveResults
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
props	I	CapeVariant (String Array)	Properties to be removed. NULL to remove all properties.

SetIndependentVar

Description

`SetIndependentVar` sets the independent variable for a given material object.

Interface Name	ICapeThermoMaterialObject
Method Name	SetIndependentVar
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
indVars	I	CapeVariant (String Array)	Independent variables to be set (see names for state variables for list of valid variables)
values	I	CapeVariant (Double Array)	Values of independent variables.

SetProp

Description

`SetProp` sets the property values of the material object.

Interface Name	ICapeThermoMaterialObject
Method Name	SetProp
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
property	I	CapeString	The property for which the values need to be set.
phase	I	CapeString	Phase, if applicable. Use NULL for a placeholder.
compIds	I	CapeVariant (String Array)	Components for which values are to be set. NULL to specify all components in the material object. For mixture property such as liquid enthalpy, this qualifier is not required. Use NULL as placeholder.
calcType	I	CapeString	The calculation type. Valid calculation types include: Pure Mixture
basis	I	CapeString	Qualifies the basis (mole / mass).
values	I	CapeVariant (Double Array)	Values to set for the property.

ValidityCheck

Description

ValidityCheck checks the validity of the calculation.

Interface Name	ICapeThermoMaterialObject
Method Name	ValidityCheck
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
props	I	CapeVariant (String Array)	The properties for which reliability is checked.
*rellist	O	CapeArrayThermoReliability	Returns the reliability scale of the calculation.

Physical Property System

The property system owns the property packages, and the system of interest must be registered. The Aspen Plus physical property system is registered as AtCOProperties.COPropertySystem.*n* where *n* is the internal version number of Aspen Plus. This number may be seen as "(n.x.y)" (where x and y are other numbers that should be ignored) after the usual version number in the **About** box.

ICapeThermoSystem Interface Methods

The ICapeThermoSystem interface consists of the following methods:

Method Name	Description
GetPropertyPackages	Returns <code>StringArray</code> of property package names supported by the system.
ResolvePropertyPackage	Resolves referenced property packages to a property package interface.

GetPropertyPackages

Description

`GetPropertyPackages` returns `StringArray` of property package names supported by the Thermodynamic System.

Interface Name	ICapeThermoSystem
Method Name	GetPropertyPackages
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*propertyPackageList	O	CapeArrayString	The returned set of supported property packages.

ResolvePropertyPackage

Description

`ResolvePropertyPackage` resolves referenced property package to a property package interface.

Interface Name	ICapeThermoSystem
Method Name	ResolvePropertyPackage
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
propertyPackage	I	CapeString	The property package to be resolved.
*propPackObject	O	CapeInterface	The property package interface.

Property Package

The property package performs requested property calculations, and contains all necessary property information, such as:

- Components.
- Property methods.
- Property data.
- The calculation methods.

The property package uses data in the material object to calculate the properties requested in the interface method. The calculated properties are then stored in the material object.

Importing and Exporting

Aspen Plus can prepare, export, and import CAPE-OPEN compliant property packages compatible with other applications.

Importing a Property Package

A CAPE-OPEN compliant property package from another application can be imported and used in Aspen Plus to calculate physical properties instead of the native physical property methods.

To import a property package:

- 1 From the **Tools** menu, select Import CAPE-OPEN Property Package. The Available Property Packages dialog box appears.
- 2 Expand the property system tree to display all the available property packages owned by a given property system.
- 3 Select the property package being imported.
- 4 Click **OK**. The property method from the foreign property package is named CP-# (where # is a number) in the **Properties | CAPE-OPEN Packages** folder. You can rename these packages if desired.

Note: If there are component conflicts, Aspen Plus prompts you to reconcile the conflict before continuing.

Details of imported packages will be displayed within the **CAPE-OPEN Packages** folder under **Customize** in the navigation pane. For packages not produced by Aspen Plus or Aspen Properties, you can choose to override the flash calculation from the package with the Aspen Properties flash. You can also specify defaults for the property method, Henry components, etc. used for any capabilities the CAPE-OPEN property package does not supply. (For packages produced by Aspen Plus or Aspen Properties, these fields are dimmed. Such packages always include all capabilities used in the Aspen Physical Property System.)

To use an imported Property Package in a simulation select the corresponding Property Method where appropriate. For example to use a CAPE-OPEN

Property Package globally select the corresponding **Property Method** on the **Global** sheet of the **Methods | Specifications** form. To use a CAPE-OPEN Property Package for a particular property analysis, select the corresponding method on the **Properties** sheet of the **Input** form for the analysis. Note that when you do so, the Henry Components, Free-Water method, etc. are dimmed; the ones in the package (or the defaults specified on the **CAPE-OPEN Packages** form for this package, if the package is missing any capabilities) are used when the method from the CAPE-OPEN package is selected.

Exporting a Property Package

Use Aspen Plus to prepare a CAPE-OPEN compliant property package for use with other applications, such as another process simulator or an in-house program.

To export a property package:

- 1 Select the components, property methods, and provide all the necessary property data and parameters.
- 2 From the **File** menu, select **Export | CAPE-OPEN Package**. The **AspenTech CAPE-OPEN Property Package Manager** window appears.
- 3 Update the description of the package as required, then click **Save and Register**.
- 4 Choose a filename and location for the CAPE-OPEN property package file.
- 5 A message appears telling you the package was successfully registered. Click **OK** to return to Aspen Plus.

ICapeThermoPropertyPackage Interface Methods

The `ICapeThermoPropertyPackage` interface consists of the following methods:

Method Name	Description
<code>GetPropertyPackages</code>	Returns <code>StringArray</code> of property package names supported by the system.
<code>ResolvePropertyPackage</code>	Resolves referenced property packages to a property package interface.

CalcEquilibrium

Description

`CalcEquilibrium` calculates and flash calculation requests.

Interface Name	<code>ICapeThermoPropertyPackage</code>
Method Name	<code>CalcEquilibrium</code>
Returns	<code>CapeError</code>

Argument(s)

Name	I/O*	Type/Dimension	Description
*materialObjectI		CapeInterface	The MaterialObject
flashType	I	CapeString	Flash calculation type.
props	I	CapeVariant (String Array)	Properties to be calculated at equilibrium. NULL for no properties.

CalcProp

Description

`CalcProp` performs all calculations, and is implemented by the associated Thermodynamic System.

Note: This method is further defined in the descriptions of the CAPE-OPEN Calling Pattern and the *User Guide Section*.

Interface Name	ICapeThermoPropertyPackage
Method Name	CalcProp
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*materialObjectI		CapeInterface	The material object for the calculation.
props	I	CapeVariant (String Array)	The list of properties to be calculated.
phases	I	CapeVariant (String Array)	List of phases for which the properties are to be calculated.
calcType	I	CapeString	Type of calculation: Mixture property or pure component property. For partial property, such as fugacity coefficients of components in a mixture, use "Mixture" CalcType. For pure component fugacity coefficients, use "Pure" CalcType.

GetComponentConstant

Description

`GetComponentConstant` returns the values of the component constants on the material object.

Interface Name	ICapeThermoPropertyPackage
Method Name	GetComponentConstant
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*materialObjectI		CapeInterface	The material object.
props	I	CapeVariant (String Array)	The list of properties.
*propvals	O	CapeArrayDouble	Component constant values.

GetComponentList

Description

`GetComponentList` returns the list of components of a given property package.

Interface Name	ICapeThermoPropertyPackage
Method Name	GetComponentList
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*compsIds	O	CapeVariant (String Array)	List of component IDs.
*formulae	O	CapeVariant (String Array)	List of component aliases.
*name	O	CapeVariant (String Array)	List of component names.
*boilTemps	O	CapeVariant (Double Array)	List of boiling point temperatures.
*molwt	O	CapeVariant (Double Array)	List of molecular weight.
*casno	O	CapeVariant (String Array)	List of CAS number.

GetPhaseList

Description

`GetPhaseList` provides the list of the supported phases.

Interface Name	ICapeThermoPropertyPackage
Method Name	GetPhaseList
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*phases	O	CapeArrayString	The list of phases supported by the property package.

GetPropList

Description

`GetPropList` returns a list of Thermodynamic System supported properties.

Interface Name	ICapeThermoPropertyPackage
Method Name	GetPropList
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*props	O	CapeArrayString	String list of all supported properties.

GetUniversalConstant

Description

`GetUniversalConstant` returns the values of the universal constants.

Interface Name	ICapeThermoPropertyPackage
Method Name	GetUniversalConstant
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*materialObjectI		CapeInterface	The material object.
props	I	CapeVariant (String Array)	List of requested universal constants.
*propvals	O	CapeArrayDouble	Values of universal constants.

PropCheck

Description

`PropCheck` verifies that properties can be calculated.

Interface Name	ICapeThermoPropertyPackage
Method Name	PropCheck
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*materialObjectI		CapeInterface	The material object for the calculations.
props	I	CapeVariant (String Array)	List of properties to check.
*valid	O	CapeArrayBoolean	The array of booleans for each property.

ValidityCheck

Description

ValidityCheck checks the validity of the calculation.

Interface Name	ICapeThermoPropertyPackage
Method Name	ValidityCheck
Returns	CapeError

Argument(s)

Name	I/O*	Type/Dimension	Description
*materialObjectI		CapeInterface	The material object for the calculations.
Props	I	CapeVariant (String Array)	The list of properties to check.
*rellist	O	CapeArrayThermoReliability	The properties for which reliability is checked.

Registration of CAPE-OPEN Components

The CAPE-OPEN components are registered using the MS Windows registry categories. The following table displays the GUIDs of all CAPE-OPEN components related to the Thermodynamic interfaces.

Category ID (CATID)	Number
CapeThermoSystem	678c09a3-7d66-11d2-a67d-00105a42887f
CapeThermoPropertyPackage	678c09a4-7d66-11d2-a67d-00105a42887f
CapeThermoEquilibriumServer	678c09a6-7d66-11d2-a67d-00105a42887f

Interface ID (IID)	Number
ICapeThermoCalculationRoutine	678c0991-7d66-11d2-a67d-00105a42887f
ICapeThermoReliability	678c0992-7d66-11d2-a67d-00105a42887f
ICapeThermoMaterialTemplate	678c0993-7d66-11d2-a67d-00105a42887f
ICapeThermoMaterialObject	678c0994-7d66-11d2-a67d-00105a42887f
ICapeThermoSystem	678c0995-7d66-11d2-a67d-00105a42887f
ICapeThermoPropertyPackage	678c0996-7d66-11d2-a67d-00105a42887f
ICapeThermoEquilibriumServer	678c0997-7d66-11d2-a67d-00105a42887f

28 COM Interface for Updating Oil Characterizations and Petroleum Properties

This chapter describes Component Object Model (COM) interfaces, and the methods available for updating pseudocomponent characterization parameters and petroleum properties during the simulation.

Note: The `IAssayUpdate` COM interface is the only interface used to update oil characterizations and properties.

Aspen Plus includes an example COM Model that uses the interface defined in this chapter. Refer to this example for the source code.

IAssayUpdate Interface Methods

The following table summarizes the main methods available for the COM interface IAssayUpdate:

Method	Descriptions
UpdateParameters	Recalculates characterization parameters for pseudocomponents.
SetPetroPropValues	Sets value for a Petro-Prop in a given stream.
CopyAssayData	Copies assay data from one material stream to another material stream.

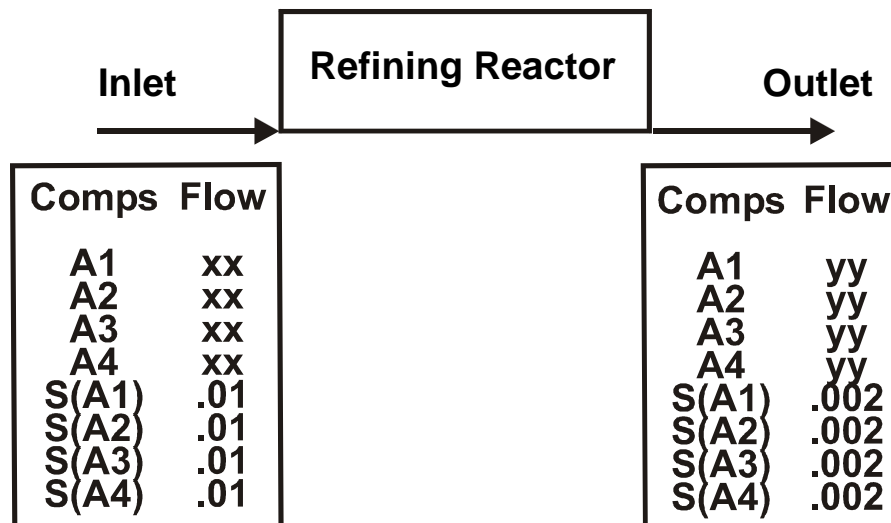
Note: There are some additional methods described further in this chapter.

The remainder of this chapter describes how to use the COM interface methods.

Modifying Petroleum Properties During a Simulation

Aspen Plus allows petroleum properties to change for the same set of pseudocomponents from stream to stream. This feature allows the simulation to track stream property changes without introducing additional components.

Consider the following example:



Both the inlet stream and outlet stream of the block have the same set of pseudocomponents. The sulfur content (S), a petroleum property of each component, is different in the two streams.

CopyAssayData

Description

Use to initialize the petroleum property data structure for the outlet stream and to copy the petroleum property information from the inlet stream to the outlet stream.

If the user model does not change the pseudocomponent petroleum property of the outlet, call this interface method for the outlet to propagate the petroleum property information from the inlet to the outlet.

Interface Name	IAssayUpdate
Method Name	CopyAssayData
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
CopyFlag	I	CopyOption	Option to use when copying. CopyCannotChange = 1: OutletStreamId references the same petroleum property data structure as the InletStreamId. The petroleum property value can not be modified. This is sufficient if the user model does not change the pseudocomponent petroleum properties. CopyCanChange = 2: the OutletStreamId has its own petroleum property data structure. Initially, this new data structure will have an exact copy of the petroleum property data as the InletStreamId. The user can modify the petroleum properties.
InletStreamId	I	BSTR	Id of the inlet stream from which the petroleum property data are copied into the outlet.
NumInletStreams	I	Short	Total number of inlet streams to the block.
InletStreamIds	I	Pointer to SAFEARRAY(BSTR)	Ids of all inlet streams.
OutletStreamId	I	BSTR	Id of the outlet stream for which the petroleum property data structure is to be setup.
FlowIn	I	Double	Total flow rate of the inlet stream referenced by InletStreamId.

Name	I/O*	Type/Dimension	Description
MissingValuesDefaultOption	I	Short	<p>Defaulting option when petroleum property in the inlet stream have missing value; used for CopyOption = CopyCanChange.</p> <p>When MissingValuesDefaultOption is 0, petroleum properties are copied from inlet to outlet without modifications, including missing values.</p> <p>When MissingValuesDefaultOption is greater than 0, components with component index greater than or equal to MissingValuesDefaultOption will default missing petroleum property to the first available value.</p>

SetPetroPropValues

Description

`SetPetroPropValues` selectively changes pseudocomponent petroleum property values for one or more pseudocomponents.

Interface Name	IAssayUpdate
Method Name	SetPetroPropValues
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
OutletStreamId	I	BSTR	The outlet stream Id associated with the petroleum property values being changed.
PropName	I	BSTR	The petroleum property name associated with the pseudocomponent values being changed (e.g., SULFUR).
NumComponents	I	Short	The pseudocomponent(s) number(s) associated with the properties being changed.
ComponentIndex	I	Pointer to SAFEARRAY(long)	Indices of pseudocomponents. Can be obtained using the GetComponentIndex method.
PetroPropValues	I	Pointer to SAFEARRAY(double)	New values of the changed petroleum property.
Temperature	I	Double	Reference temperature of the data. Used for viscosity or kinematic viscosity.
Unit	I	BSTR	The units of the reference temperature (one of K, C, R, or F).

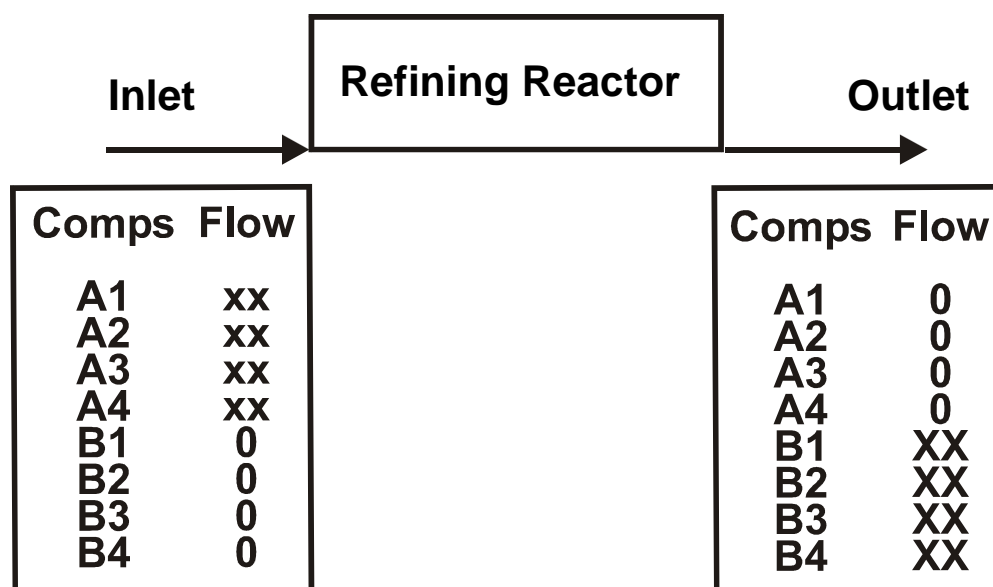
Recalculate Characterization Parameters

Pseudocomponent characterization parameters (e.g., critical temperature, critical pressure, ideal gas heat capacity parameters, vapor pressure Extended-Antoine constants, liquid viscosity parameters, etc.) are not subject to change during the simulation.

If a parameter such as the pseudocomponent molecular weight changes during a simulation, a material imbalance may occur.

Conversely, in modeling petroleum refining reactor units, boiling point and gravity data of the reactor outlet stream are calculated during the simulation. The pseudocomponent characterization parameters must then be re-computed based on these calculations.

Consider the following example:



The simulation contains 2 sets of pseudocomponents:

- (A1 ... A4)
- (B1 ... B4)

The inlet stream contains flows for the "A" set and the outlet stream contains flows for the "B" set. The property constants of the "B" set have to be re-computed during the simulation, based on the boiling point and gravity values predicted for the outlet stream by the reactor model.

UpdateParameters

Description

UpdateParameters updates the pseudocomponent properties during the simulation.

Interface Name	IAssayUpdate
Method Name	UpdateParameters
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
NumPseudoComponents	I	short	Number of pseudocomponents.
ComponentIndex	I	Pointer to SAFEARRAY(long)	Component indices of pseudocomponents.
Note: Do not use pseudocomponents that appear in any flowsheet inlet stream with a non-zero flow.			
PseudoCompDefiningParamValues	I	Pointer to SAFEARRAY(double)	Values of parameters used to recalculate the characterization parameters: TB, API, SG, and MW. The length of the array is 4*NumPseudoComponents. The first four elements store TB, API, SG, and MW of the pseudocomponent corresponding to ComponentIndex[0]; and the next four elements store TB, API, SG, and MW of the pseudocomponent corresponding to ComponentIndex[1], and so on.
ReportFlag	I	ReportOption	Report flag. Currently not active.

Note: Pseudocomponents can appear in a flowsheet inlet stream as themselves, in assay form, or in a blend. If an assay or blend has a non-zero flow in a flowsheet inlet stream, then all associated pseudocomponents are considered as appearing in the flowsheet inlet stream with a non-zero flow. In Aspen Plus, pseudocomponents are generated based on specifications of a PC-CALC paragraph. All assays and blends in a PC-CALC paragraph share the same set of pseudocomponents. If no PC-CALC paragraph is specified in a flowsheet, then all assays and blends in the flowsheet share one (and only one) set of pseudocomponents.

Additional IAssayUpdate Interface Methods

The following table summarizes additional methods available for the COM interface IAssayUpdate:

Method	Descriptions
GetAssayCount	Returns number of user-entered assays and blends.
GetAssayId	Returns an Assay/Blend Id, given index.
GetComponentCount	Returns the number of light-end components and pseudocomponents.
GetComponentIndex	Returns component indices for all light-end components and pseudocomponents associated with the assay.
GetParameters	Returns parameters for pseudocomponents.
GetPetroPropValues	Returns values of petroleum properties in a given stream.
GetPropertyCount	Returns number of user-entered petroleum properties.
GetPropertyName	Returns petroleum property Id, given petroleum property index.

GetAssayCount

Description

GetAssayCount returns the number of assays and blends defined in a simulation.

Interface Name	IAssayUpdate
Method Name	GetAssayCount
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
Count	O	Pointer to Int	Number of assays and blends.

GetAssayId

Description

GetAssayId returns the assay/blend Id given the index.

Interface Name	IAssayUpdate
Method Name	GetAssayId
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
Index	I	Int	Assay/blend index.
AssayId	O	Pointer to BSTR	Assay/blend Id.

GetComponentCount

Description

`GetComponentCount` returns the number of light-end components and pseudocomponents that exist in an assay/blend.

Interface Name	IAssayUpdate
Method Name	GetComponentCount
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
AssayId	I	BSTR	The assay/blend Id.
NumLightEndComponents	O	Pointer to short	The number of light end components in the assay/blend.
NumPseudoComponents	O	Pointer to short	The number of pseudocomponents in the assay/blend.

GetComponentIndex

Description

`GetComponentIndex` returns pseudocomponent indices given an assay/blend Id. This is necessary when a user defines the inlet stream and outlet stream in terms of 2 assays/blends, rather than in terms of the individual pseudocomponents. In this scenario, the pseudocomponent indices are not known a priori.

Interface Name	IAssayUpdate
Method Name	GetComponentIndex
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
AssayId	I/O	Pointer to BSTR	The assay/blend Id.
NumLightEndComponents	I/O	Pointer to short	The number of light end components in the assay/blend.
NumPseudoComponents	I/O	Pointer to short	Number of pseudocomponents in the assay/blend.
LightEndComponentIndex	I/O	Pointer to SAFEARRAY(long)	Component indices of the light end components.
PseudoComponentIndex	I/O	Pointer to SAFEARRAY(long)	Component indices of the pseudocomponents.

GetParameters

Description

`GetParameters` returns characterization parameters for the components.

Interface Name	IAssayUpdate
Method Name	GetParameters
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
ParaName	I/O	BSTR	The parameter name.
NumPseudoComponents	I	Short	The number of pseudocomponents.
PseudoComponentIndex	I/O	Pointer to SAFEARRAY(long)	Component indices of the pseudocomponents.
PseudoComponentParamValues	I/O	Pointer to SAFEARRAY(double)	The pseudocomponent parameter values.

GetPetroPropValues

Description

`GetPetroPropValues` returns values of a petroleum property defined for a given stream.

Interface Name	IAssayUpdate
Method Name	GetPetroPropValues
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
OutletStreamId	I	BSTR	The outlet stream Id.
PropName	I	BSTR	The petroleum property name (e.g., SULFUR).
NumComponentIndex	I	Short	The number of pseudocomponent(s).
ComponentIndex	I	Pointer to SAFEARRAY (long)	Indices of pseudocomponents. This can be obtained using the GetComponentIndex method.
PetroPropValues	O	Pointer to SAFEARRAY (double)	The values of the petroleum property.
Temperature	I	Double	The reference temperature of the property. Used for viscosity or kinematic viscosity.
Unit	I	BSTR	The units of the reference temperature (one of K, C, R, or F).

GetPropertyCount

Description

`GetPropertyCount` returns the number of petroleum properties defined in a simulation.

Interface Name	IAssayUpdate
Method Name	GetPropertyCount
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
Count	O	Pointer to Int	Number of petroleum properties.

GetPropertyName

Description

`GetPropertyName` returns the name of the petroleum property given the index.

Interface Name	IAssayUpdate
Method Name	GetPropertyName
Returns	Success/failure flag

Argument(s)

Name	I/O*	Type/Dimension	Description
Index	I	Int	Petroleum property index.
Name	O	Pointer to BSTR	Petroleum property name.

A Common Blocks and Accessing Component Data

The first section of this appendix contains descriptions of the Aspen Plus Fortran common blocks that you can use in user subroutines. The second section contains information on how to access certain component data areas in DMS_PLEX.

Aspen Plus Common Blocks

The user routine should not change any of the variables in the commons, except for COMMON /DMS_PLEX/.

This section describes the following common blocks:

Common Block	Description
DMS_ERROUT	Error message buffer
DMS_FLSCOM	Flash error common
DMS_NCOMP	Number of components
DMS_PLEX	The Plex
PPUTL_PPGLOB	Physical property global variables
DMS_RGLOB	Real global variables
DMS_RPTGLB	Report writer global variables
DMS_STWKWK	Stream flash work space
SHS_STWORK	Stream flash work space offsets
PPEXEC_USER	Control flags and other variables for user-written subroutines

COMMON DMS_ERROUT

DMS_ERROUT contains character variables for error messages (see Chapter 4, Aspen Plus Error Handler). To use DMS_ERROUT, include the following directive (in column 1) in your subroutine:

```
#include "dms_errout.cmn"
```

The following table describes the variables in DMS_ERROUT.

Variable	Type	Description
ERROUT_IEROUT	CHARACTER*80	Character array of length 10 used to store error messages. See Chapter 4, Aspen Plus Error Handler.

COMMON DMS_FLSCOM

DMS_FLSCOM contains error return codes from lower level flash routines, to be checked by higher level flash routines. To use DMS_FLSCOM, include the following directive (in column 1) in your subroutine:

```
#include "dms_flscm.cmn"
```

The following table describes the variables in DMS_FLSCOM.

Variables	Description
FLSCOM_IFLERR	Return code being passed up from lower level routines: 0 = all OK 1 = error encountered

COMMON DMS_NCOMP

DMS_NCOMP contains numbers of components of various classifications and lengths of component-related stream segments. To use DMS_NCOMP, include the following directive (in column 1) in your subroutine:

```
#include "dms_ncomp.cmn"
```

The following table describes the variables in DMS_NCOMP.

Variable	Description
NCOMP_NCC	Number of conventional components defined by user + components generated by ADA
NCOMP_NNCC	Number of nonconventional components
NCOMP_NC	Total number of components = NCOMP_NCC + NCOMP_NNCC
NCOMP_NAC	Number of attributed components
NCOMP_NACC	Number of attributed conventional components
NCOMP_NVCP	Number of conventional substream variables without attributes (NCOMP_NCC+9)
NCOMP_NVNCP	Number of nonconventional substream variables without attributes (NCOMP_NNCC+9)
NCOMP_NVACC	Length of the component attribute set for all attributed conventional components
NCOMP_NVANCC	Length of the component attribute set for all nonconventional components
NCOMP_NASSAY	Number of assays
NCOMP_NCCASS	Number of conventional components defined by user plus number of assays = NCOMP_NCC + NCOMP_NASSAY
NCOMP_IDXWAT	Component index number for water Note: This indicates the first component with alias H2O if there is more than one. Use other functions to search the component list if you require compatibility with specifying multiple components with the same alias.
NCOMP_NG	Number of UNIFAC groups
NCOMP_NM	Number of molecular species
NCOMP_NAION	Number of anions
NCOMP_NCION	Number of cations
NCOMP_NPOLY	Number of polymer components (Aspen Polymers)
NCOMP_NSEG	Number of segment components (Aspen Polymers)
NCOMP_NOLIG	Number of oligomer components (Aspen Polymers)
NCOMP_NSITE	Number of sites (Aspen Polymers)
NCOMP_NCAT	Number of Ziegler-Natta catalyst components (Aspen Polymers)
NCOMP_NINIT	Number of ionic polymerization initiator components (Aspen Polymers)

COMMON DMS_PLEX

DMS_PLEX contains the lengthy in-core storage area for the Plex. The Plex is the main memory area where all the data for a simulation is stored, in a

flexible way. The Plex consists of data areas sized to hold the amount of data in the simulation. See Accessing Component Data Using the Plex, later in this chapter, for more information.

The Plex consists of integer and real arrays equivalenced to each other. The length depends on problem size and can change (grow) during program execution.

To use DMS_PLEX, include the following directive (in column 1) in your subroutine:

```
#include "dms_plex.cmn"
```

Add the following declarations to your subroutine:

```
REAL*8 B(1)
EQUIVALENCE (B(1), IB(1))
```

The following table describes the variables in DMS_PLEX.

Variable	Description
B	Real Plex area
IB	Integer Plex area

COMMON PPUTL_PPGLOB

Physical property global common used by all simulation program property routines for easy reference. To use PPUTL_PPGLOB, include the following directive (in column 1) in your subroutine:

```
#include "pputl_ppglob.cmn"
```

The following table describes the variables in PPUTL_PPGLOB.

Variable	Description
PPGLOB_PREF	Reference pressure (101,325 N/m ²)
PPGLOB_TREF	Reference temperature (298.15 K)
PPGLOB_RGAS	Gas law constant (8,314.33 J/kgmole-K)
PPGLOB_BOLTZ	Boltzmann constant (1.38048x10 ⁻²³ J/K)

COMMON DMS_RGLOB

DMS_RGLOB contains real global specifications. To use DMS_RGLOB, include the following directive (in column 1) in your subroutine:

```
#include "dms_rglob.cmn"
```

The following table describes the variables in DMS_RGLOB.

Variable	Description
RGLOB_RMISS	Missing code for real numbers, 1D35
RGLOB_RMIN	Smallest nonzero magnitude for a real number, 1D-15
RGLOB_ABSMIN	Smallest allowed absolute tolerance, 1D-8
RGLOB_SCLMIN	Smallest allowed scale factor, 1D-10 (not currently used)
RGLOB_XMIN	Smallest mole fraction considered nonzero, 1D-15
RGLOB_HSCALE	Scale factor for enthalpy, 1D8
RGLOB_RELMIN	Smallest allowed relative tolerance, 1D-10 (not currently used)
RGLOB_SCLDEF	Default scale factor, 1D8 (currently used for pressure)
RGLOB_TMAX	Time control parameter - maximum time in simulation allowed
RGLOB_TNOW	Time control parameter - current time in simulation
RGLOB_TUPPER	Upper limit for temperature (from Setup SimulationOptions FlashConvergence sheet)
RGLOB_TLOWER	Lower limit for temperature (from Setup SimulationOptions FlashConvergence sheet)
RGLOB_PUPPER	Upper limit for pressure (from Setup SimulationOptions FlashConvergence sheet)
RGLOB_PLOWER	Lower limit for pressure (from Setup SimulationOptions FlashConvergence sheet)

COMMON DMS_RPTGLB

DMS_RPTGLB contains report writer flags and space for subsection headings. To use DMS_RPTGLB, include the following directive (in column 1) in your subroutine:

```
#include "dms_rptglb.cmn"
```

The following table describes the variables in DMS_RPTGLB.

Variable	Description
RPTGLB_IREPFL	Report pass flag 0 - not report pass; 1 - report pass
RPTGLB_ISUB(10)	Subsection heading set by Report Writer routines and passed to routine RPTHDR
RPTGLB_IRGSUM	Summary file flag. 0 = no summary file requested; 1=summary file requested

COMMON DMS_STWKWK

DMS_STWKWK is the stream flash work area common. This is a work common designed for stream flash calculations. It is always used along with COMMON SHS_STWORK. COMMON SHS_STWORK sets the dimension and offset of various work areas in the COMMON DMS_STWKWK.

To use DMS_STWKWK, include the following directive (in column 1) in your subroutine:

```
#include "dms_stwkwk.cmn"
```

The following table describes the variables in DMS_STWKWK.

Variable	Description
STWKWK_LRSTW	Offset into the real work area in the Plex. B(STWKWK_LRSTW+1) is the first position in the Plex
STWKWK_LISTW	Offset into the integer work area in the Plex. IB(STWKWK_LISTW+1) is the first position in the Plex
STWKWK_NCPMOO	Number of packed components in MIXED substream
STWKWK_NCPCSO	Number of packed components in all CISOLID substreams combined
STWKWK_NCPNCO	Number of packed components in all NC substreams combined
STWKWK_NTRIAL	Number of iterations
STWKWK_TCALC	Outlet temperature (K)
STWKWK_PCALC	Outlet pressure (N/m ²)
STWKWK_VCALC	Outlet molar vapor fraction
STWKWK_QCALC	Heat duty (watt)
STWKWK_BETA	Molar ratio of liquid 1 to total liquid

COMMON SHS_STWORK

SHS_STWORK is the stream flash work offset COMMON. This common contains indexes and dimension for the stream flash work COMMON DMS_STWKWK. To use SHS_STWORK, include the following directive (in column 1) in your subroutine:

```
#include "shs_stwork.cmn"
```


The following table describes the variables in SHS_STWORK.

Variable	Description
STWORK_NRETN	Dimension of retention real vector
STWORK_NIRETN	Dimension of retention integer vector
STWORK_MF	Offset to overall MIXED substream (feed) mole fraction vector Mole fraction vector begins at B(STWKWK_LRSTW + STWORK_MF) Allocated length is NCOMP_NCC
STWORK_MX	Offset to overall liquid mole fraction vector Mole fraction vector begins at B (STWKWK_LRSTW + STWORK_MX) Allocated length is NCOMP_NCC
STWORK_MX1	Offset to liquid 1 mole fraction vector Mole fraction vector begins at B (STWKWK_LRSTW + STWORK_MX1) Allocated length is NCOMP_NCC
STWORK_MX2	Offset to liquid 2 mole fraction vector Mole fraction vector begins at B (STWKWK_LRSTW + STWORK_MX2) Allocated length is NCOMP_NCC
STWORK_MY	Offset to vapor mole fraction vector Mole fraction vector begins at B (STWKWK_LRSTW + STWORK_MY) Allocated length is NCOMP_NCC
STWORK_MCS	Offset to conventional solid mole fraction vector Mole fraction vector begins at B (STWKWK_LRSTW + STWORK_MCS) Allocated length is NCOMP_NCC
STWORK_MNC	Offset to nonconventional solid mass fraction vector Mole fraction vector begins at B (STWKWK_LRSTW + STWORK_MNC) Allocated length is NCOMP_NNCC
STWORK_MRETN	Offset to retention real vector Retention vector begins at B(STWKWK_LRSTW + STWORK_MRETN) Length is STWORK_NRETN
STWORK_MIM	Offset to IDX vector for MIXED substream IDX vector begins at IB(STWKWK_LISTW+ STWORK_MIM) Allocated length is NCOMP_NCC
STWORK_MIC	Offset to IDX vector for conventional solids IDX vector begins at IB(STWKWK_LISTW + STWORK_MIC) Allocated length is NCOMP_NCC
STWORK_MIN	Offset to IDX vector for nonconventional solids IDX vector begins at IB (STWKWK_LISTW + STWORK_MIN) Allocated length is NCOMP_NNCC
STWORK_MIRETN	Offset to retention integer vector Retention vector begins at B(STWKWK_LITW + STWORK_MIRETN) Length is STWORK_NIRETN
STWORK_HV	Vapor enthalpy (J/kgmole)
STWORK_HL	Liquid enthalpy (J/kgmole)
STWORK_HL1	Liquid 1 enthalpy (J/kgmole)
STWORK_HL2	Liquid 2 enthalpy (J/kgmole)
STWORK_SV	Vapor entropy (J/kgmole-K)
STWORK_SL	Liquid entropy (J/kgmole-K)
STWORK_SL1	Liquid 1 entropy (J/kgmole-K)
STWORK_SL2	Liquid 2 entropy (J/kgmole-K)
STWORK_VV	Vapor volume (m ³ /kgmole)

Variable	Description
STWORK_VL	Liquid volume (m ³ /kgmole)
STWORK_VL1	Liquid 1 volume (m ³ /kgmole)
STWORK_VL2	Liquid 2 volume (m ³ /kgmole)
STWORK_XMWV	Vapor molecular weight
STWORK_XMWL	Liquid molecular weight
STWORK_XMWL1	Liquid 1 molecular weight
STWORK_XMWL2	Liquid 2 molecular weight
STWORK_HCS	CISOLID enthalpy (J/kgmole)
STWORK_HNCS	NC enthalpy (J/kg)
STWORK_SSALT	Calculated output salt entropy (J/kgmole-K)
STWORK_VSALT	Calculated output salt volume (m ³ /kgmole)
STWORK_HSALT	Calculated output salt enthalpy (J/kgmole)
STWORK_FSALT	Mole ratio of total salt to total salt-free feed
STWORK_RATIO	Mole ratio of product versus reactant

COMMON PPEXEC_USER

PPEXEC_USER conveniently stores run time control flags. To use PPEXEC_USER, include the following directive (in column 1) in your subroutine:

```
#include "ppexec_user.cmn"
```

The following table describes the variables in PPEXEC_USER.

Variable	Description
USER_RUMISS	Real missing code
USER_IUMISS	Integer missing code
USER_NGBAL	Local energy balance flag: 0 = OFF 1 = ON
USER_IPASS	Calculation control flag (for User and User2 only): 1 = Perform simulation calculations only 2 = Perform results calculations only 3 = Perform simulation and results calculations 4 = Write report
USER_IRESTR	Local restart flag: 0 = Initialization calculations should be performed 1 = Initialization calculations should be performed if model calculations are successful then USER_IRESTR should be set to 2, otherwise left at 1 2 = Initialization should be bypassed. If model calculations are successful leave at 2, otherwise reset to 1.
USER_ICONVG	Local convergence flag (for User and User2 only): 0 = Calculations completed normally - I = Calculations failed. I indicates the reason
USER_LMSG	Local diagnostic flag
USER_LPMSG	Local physical property diagnostic flag

Variable	Description
USER_NHSTRY	History file Fortran unit number
USER_NRPT	Report file Fortran unit number
USER_NTRMNL	Terminal file Fortran unit number
USER_ISIZE	Sizing calculation flag (for User and User2 only) 0 = No sizing calculations (default) 1 = Do sizing calculations
USER_BALMAS	Whether Aspen Plus should perform a mass balance check on completion of user unit operation model and report warnings on mass balance errors 0 = Perform mass balance check (default) -1 = Do not perform mass balance check

Accessing Component Data Using the Plex

All the data for conventional and nonconventional components are stored in the labeled common DMS_PLEX. Each type of data occupies a contiguous area within the Plex. These areas are of variable lengths depending on the amount of data within the simulation. To use the Plex, the user will need to know:

- The name of the area.
- The offset of the area into the Plex.
- The structure of that area (one- or two-dimensional; and if two-dimensional, the leading dimension).

You can easily obtain the offset into the Plex by using the utility DMS_IFCMNC (see Chapter 4). The data for two-dimensional areas are stored in columns.

This section describes the following component data areas:

Data Area	Description
FRMULA	Conventional component alias
IDSCC	Conventional component IDs
IDSNCC	Nonconventional component IDs
IDXNCC	Nonconventional attributed component indexes
paramname	Any physical property parameter
Using IPOFF3	Some calculated or intermediate properties

FRMULA

FRMULA is the conventional component alias stored as three integer words for each conventional component (equivalent to 12 characters). Its length is 3*NCOMP_NCC.

Example: Printing the Alias of the Ith Component to the History File

```
#include "dms_plex.cmn"
#include "ppexec_user.cmn"
      INTEGER DMS_IFCMNC
      .
      .
      .
      LFRMUL = DMS_IFCMNC('FRMULA')
      LI = LFRMUL + 3*(I-1)
      WRITE (USER_NHSTRY, '(6X, A, 3A4)') 'ALIAS = ',
      *      (IB(LI+J), J=1,3)
```

Note: This WRITE statement will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

IDSCC

IDSCC is the conventional component IDs stored as two integer words for each conventional component (equivalent to 8 characters). Its length is 2*NCOMP_NCC.

Example: Printing the ID of the Ith Conventional Component to the History File

```
#include "dms_plex.cmn"
#include "ppexec_user.cmn"
      INTEGER DMS_IFCMNC
      .
      .
      .
      LIDSCC = DMS_IFCMNC('IDSCC')
      LI = LIDSCC + 2*(I-1)
      WRITE (USER_NHSTRY, '(6X, A, 2A4)') 'COMPONENT = ',
      *      (IB(LI+J), J=1,2)
```

Note: This WRITE statement will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

IDSNCC

IDSNCC is the nonconventional component IDs stored as two integer words for each nonconventional component (equivalent to 8 characters). Its length is 2*NCOMP_NNCC.

Example: Printing the ID of the Ith Nonconventional Component to the History File

```
#include "dms_plex.cmn"
#include "ppexec_user.cmn"
      INTEGER DMS_IFCMNC
      .
```

```

      .
      .
      LIDSNC = DMS_IFCMNC('IDSNCC')
      LI = LIDSNC + 2*(I-1)
      WRITE (USER_NHSTRY, '(6X, A, 2A4)') 'COMPONENT = ',
      *      (IB(LI+J), J=1,2)

```

Note: This WRITE statement will not work when using the Compaq Visual Fortran compiler. See **Moving to the Intel Fortran Compiler**, chapter 1.

IDXNCC

IDXNCC stores the pointer into the component attribute array (CAT) for the beginning of the attributes for each attributed nonconventional component. Its length is NCOMP_NAC – NCOMP_NACC (equal to NCOMP_NNCC if all nonconventional components are attributed).

Example: Storing the First CAT Location for the Attributes of the Nonconventional Component ASH into the Variable LASH

```

#include "dms_plex.cmn"
      INTEGER DMS_KNCIDC, DMS_IFCMNC
      .
      .
      .
      KASH = DMS_KNCIDC('ASH')
      LIDXNC = DMS_IFCMNC('IDXNCC')
      LASH = IB(LIDXNC+KASH)

```

Paramname

Paramname is the physical property parameter data for the indicated parameter name.

If unary, its length is nel*ncomp

If binary, its length is nel*ncomp*ncomp

Where:

nel = Number of elements per parameter

ncomp = Number of conventional or nonconventional components
(NCOMP_NCC or NCOMP_NNCC)

Example: Locating Physical Property Data for the Ith Conventional Component

```

#include "dms_plex.cmn"
      DIMENSION B(1)
      EQUIVALENCE (B(1), IB(1))
      INTEGER DMS_IFCMNC

      DIMENSION PLXANT (9)

```

```

C**
C**  LOCATE PLEX OFFSETS
C**
      LMW = DMS_IFCMNC('MW')
      LTC = DMS_IFCMNC('TC')
      LPC = DMS_IFCMNC('PC')
      LCHRG = DMS_IFCMNC('CHARGE')
      LPL = DMS_IFCMNC('PLXANT')
C**
C**  LOCATE OFFSETS FOR COMPONENT I
C**
      LMWI = LMW+I
      LTCI = LTC+I
      LPCI = LPC+I
      LCHRG I = LCHRG+I
      LPLI = LPL+9*(I-1)
C**
C**  ACCESS PROPERTY DATA FROM THE PLEX
C**
      XMW = B(LMWI)
      TC = B(LTCI)
      PC = B(LPCI)
      CHARGE = B(LCHRG I)
      DO J = 1, 9
      PLXANT(J) = B(LPL+J)
      END DO

```

Using IPOFF3

Users familiar with the Aspen Plus system have been able to directly access some calculated or intermediate properties from the plex using the labeled common IPOFF3_IPOFF3. For example, they might access activity coefficient with the following code:

```

      INTEGER GAM, LGAMA, I
      REAL*8 GAMMA1

      GAM(I) = LGAMA + I
      LGAMA = IPOFF3_IPOFF3(24)

```

```

C  ln(gamma) of the first component in the mixture
      GAMMA1 = B(GAM(1))

```

In version 2006, a project to reduce memory usage by Aspen Plus has resulted in a major change to the way these calculated properties are stored. As a result, labeled common IPOFF3_IPOFF3 can no longer be used in this way. Instead, use the DMS_ALIPOFF3 function to determine the offset, as illustrated by the following code:

```

C  Include a new labelled common
#include "dms_lclist.cmn"

C  Declare the variables
      INTEGER DMS_ALIPOFF3, FN, PROP1, LPROP1, I, PROP2, LPROP2
      REAL*8 GAMMA1, XTRUE

C  Define the function FN

```

```

C Use this function to find the correct offset for most
C real-valued properties. Not used for XTRUE and integer
C properties
      FN(I) = I + LCLIST_LBLCLIST

C Get the plex offsets
C 24 is the property index for ln(gamma)
C 76 is the property index for xtrue
      LPROP1 = DMS_ALIPOFF3(24)
      LPROP2 = DMS_ALIPOFF3(76)

C Define function PROP1
C PROP1(I) can be used as GAM(I) was used before.
      PROP1(I) = FN(LPROP1) + I

C Access the property from the Plex B()
C ln(gamma) of the first component in the mixture
      GAMMA1 = B(PROP1(1))

C Define function PROP2
C Use this form for XTRUE and all integer-valued properties
      PROP2(I) = LPROP2 + I

C Access the property from the Plex B()
C xtrue of the first component in the mixture
      XTRUE = B(PROP2(1))

```

Accessing Component Data using PPUTL_GETPARAM

It is also possible to access component characterization parameters with a subroutine without using the commons required to access the Plex. PPUTL_GETPARAM handles scalar parameters such as TB, TC, PC; temperature-dependent parameters such as PLXANT; binary parameters such as RKSKIJ and NRTL; and Unifac group and group binary parameters.

Calling Sequence for PPUTL_GETPARAM

```

SUBROUTINE PPUTL_GETPARAM      (NAME, NCOMP, IDX, VALUE, NELEM,
                                NDIM, IERROR)

```

Argument List Descriptions for PPUTL_GETPARAM

Variable	I/O [†]	Type	Dimension	Description
NAME	I	INTEGER	2	Parameter name
NCOMP	I	INTEGER	—	Number of components or groups
IDX	I	INTEGER	NCOMP	Component index vector, non-conventional component index vector, or Unifac group index vector
VALUE	O	REAL*8	*	Parameter values (See note.)
NELEM	O	INTEGER	—	Number of elements

NDIM	O	INTEGER	—	Parameter type: 1 = Unary, conventional components 2 = Binary, conventional components 3 = Unary, non-conventional components 4 = Binary, non-conventional components 5 = Unifac group 6 = Unifac group binary
------	---	---------	---	--

IERROR	O	INTEGER	—	0 = No error 1 = Parameter not available
--------	---	---------	---	---

† I = Input to subroutine, O = Output from subroutine

Note: VALUE must be dimensioned adequately by the calling program.

Parameter Type	Size	Parameter Stack
Unary scalar	NCOMP	1 to NCOMP
Unary T-dependent	NELEM * NCOMP	Element 1 to NELEM for component 1, then all elements for component 2, etc.
Binary scalar	NCOMP * NCOMP	K(I,J) with I,J indexed as I + NCOMP * (J-1)
Binary vector	NELEM * NCOMP * NCOMP	K(1,I,J) with I,J pair indexed as for binary scalars, then K(2,I,J), etc.

To avoid complex indexing, it is recommended that the calling program dimensions VALUE as VALUE(NELEM,NCOMP) for unary T-dependent parameters, VALUE(NCOMP,NCOMP) for binary scalar parameters, and VALUE(NELEM,NCOMP,NCOMP) for binary vectors.

B User Subroutine Templates and Examples

When you start writing a new user subroutine you can use any of the templates or examples that are provided online. The directories in which these files are located are shown below. Please read the README.TXT file first, for a brief description of all the files provided in each directory.

C:\Program Files\AspenTech\Aspen Plus <version>\Engine\User

C:\Program Files\AspenTech\APrSystem <version>\Engine\User

Substitute the installation drive and directory for C:\Program Files\AspenTech if you did not install the Aspen Plus Simulation Engine or the APrSystem in the default directory.

C Stream Structure

The stream vector is a concatenation of the substream vectors of all substreams that make up the stream. Each stream belongs to a stream class, which defines the number and order of its substreams. For built-in stream classes, the order in which substreams appear in the stream vector is the order in which the substreams are listed in the following table:

Stream Class	Number of Substreams [†]	Substream Type(s) [†]	Substream ID(s)	Substream PSD ID
CONVEN	1	MIXED	MIXED	—
MIXCISLD	2	MIXED CISOLID	MIXED CISOLID	—
MIXNC	2	MIXED NC	MIXED NC	— —
MIXCINC	3	MIXED CISOLID NC	MIXED CISOLID NC	— — —
MIXCIPSD	2	MIXED CISOLID	MIXED CIPSD	— PSD
MCINCPSD	3	MIXED CISOLID NC	MIXED CISOLID NCPSD	— PSD PSD

[†] The arguments NSUBS, IDXSUB, and ITYPE contain the stream class data structure information, and are defined in all user model subroutines dealing with stream vectors.

For user-defined stream classes, the order is that in which you listed them on the **Define Stream Class** dialog box (from the **Setup | Stream Class** form).

The structure of each individual substream type is described in the following sections.

Substream MIXED

Array Index	Description
1, . . . , NCOMP_NCC	Component mole flows (kgmole/sec)
NCOMP_NCC + 1	Total mole flow (kgmole/sec)
NCOMP_NCC + 2	Temperature (K)
NCOMP_NCC + 3	Pressure (N/m ²)
NCOMP_NCC + 4	Mass enthalpy (J/kg)
NCOMP_NCC + 5	Molar vapor fraction
NCOMP_NCC + 6	Molar liquid fraction
NCOMP_NCC + 7	Mass entropy (J/kg-K)
NCOMP_NCC + 8	Mass density (kg/m ³)
NCOMP_NCC + 9	Molecular weight (kg/kgmole)
NCOMP_NCC + 10, . . . , NCOMP_NCC + 9 + NCOMP_NVACC	Component attributes

NCOMP_NCC is the number of conventional components entered on the **Components | Specifications | Selection** sheet and NCOMP_NVACC is the total length of the component attribute array for conventional components (see Appendix A, DMS_NCOMP). The order of the component mole flows is the same as the order of the components on the **Components | Specifications | Selection** sheet. All values are in the SI units indicated.

The component attributes appear in the MIXED substream mainly for polymer simulations. They appear in the same order as described for Substream NC, below.

Substream CISOLID

The layout of a substream vector for a substream of type CISOLID is shown below. The layout is the same as for a MIXED substream, except that if the substream has a PSD, an array of values for the PSD is appended to the vector. NCOMP_NCC is the number of conventional components. Space for all of the conventional components is reserved in both the MIXED and the CISOLID type substream. The component order is the same as on the **Components | Specifications | Selection** sheet. All values are in the SI units indicated. For substreams of type CISOLID, vapor and liquid fractions have the value 0.0.

Array Index	Description
1, . . . , NCOMP_NCC	Conventional component mole flows (kgmole/sec)
NCOMP_NCC + 1	Total mole flow (kg-mole/sec)
NCOMP_NCC + 2	Temperature (K)
NCOMP_NCC + 3	Pressure (N/m ²)
NCOMP_NCC + 4	Mass enthalpy (J/kg)

Array Index	Description
NCOMP_NCC + 5	Molar vapor fraction (0.0)
NCOMP_NCC + 6	Molar liquid fraction (0.0)
NCOMP_NCC + 7	Mass entropy (J/kg-K)
NCOMP_NCC + 8	Mass density (kg/m ³)
NCOMP_NCC + 9	Molecular weight (kg/kgmole)
NCOMP_NCC + 10	frac ₁
	.
	PSD values (if a PSD is defined for the substream) †
	.
NCOMP_NCC + 9 + n	frac _n

† n is the number of intervals in the particle size distribution.

Substream NC

The layout of a substream vector for a substream of type NC is shown below. In addition to the component flows and the stream conditions, the vector contains component attributes and, if the substream has a PSD, an array of values for the PSD. NCOMP_NNCC is the number of nonconventional components, and NCOMP_NVANCC is the total length of the component attribute array for nonconventional components. (see Appendix A, DMS_NCOMP).

The component order for the component flows is the same as on the **Components | Specifications | Selection** sheet. Component attributes for each component appear in the order specified on the **Components | Component Attributes | Selection** sheet or the **Methods | NC Props | Property Methods** sheet for that component. On the **NC Props** form, this order depends on the methods chosen for calculating nonconventional component properties.

All values are in the SI units indicated. n is the number of intervals in the particle size distribution. ncat is the number of elements of each component attribute. For substreams of type NC, the molecular weight is set to 1 by definition.

Array Index	Description
1, . . . , NCOMP_NNCC	Component mass flows (kg/s)
NCOMP_NNCC + 1	Total mass flow (kg/s)
NCOMP_NNCC + 2	Temperature (K)
NCOMP_NNCC + 3	Pressure (N/m ²)
NCOMP_NNCC + 4	Mass enthalpy (J/kg)
NCOMP_NNCC + 5	Vapor fraction (0.0)
NCOMP_NNCC + 6	Liquid fraction (0.0)
NCOMP_NNCC + 7	Mass entropy (J/kg-K)
NCOMP_NNCC + 8	Mass density (kg/m ³)
NCOMP_NNCC + 9	1.0

Array Index	Description
NCOMP_NNCC + 10, ... , NCOMP_NNCC + 9 + NCOMP_NVANCC	<div> <div> value₁ . . . value_{ncat1} </div> <div> } </div> <div> Values for component attribute 1 of component 1 † </div> </div>
value ₁ . . . value _{ncat2}	<div> <div> value₁ . . . value_{ncat2} </div> <div> } </div> <div> Values for component attribute 2 of component 1 † </div> </div>
...	
value ₁ . . . value _{ncat1}	<div> <div> value₁ . . . value_{ncat1} </div> <div> } </div> <div> Values for component attribute 1 of component 2 † </div> </div>
...	
NCOMP_NNCC + 10 + NCOMP_NVANCC, frac ₁ ... , NCOMP_NNCC + 9 + NCOMP_NVANCC + n frac _n	<div> <div> NCOMP_NNCC + 10 + NCOMP_NVANCC, frac₁ ... , NCOMP_NNCC + 9 + NCOMP_NVANCC + n frac_n </div> <div> } </div> <div> PSD values (if a PSD is defined for the substream) †† </div> </div>

† ncat1 is the number of elements of the first component attribute.
ncat2 is the number of elements of the second component attribute.
†† n is the number of intervals in the particle size distribution.

Determining Particle Size Distribution Length

Use SHS_LOCPSD to determine the number of size intervals in the particle size distribution for any substream, given the stream vector and the stream class descriptor bead LD. In User3 LD is provided as LDMAT. See [General Stream Handling Utilities](#) in Chapter 4.