WHEN YOU NEED TO BE SURE

CyFlex® Composition and Property Variables

Version 8

June 20, 2022

Developed by SGS North America, Inc.

Version History

Version	Date	Revision Description
1	1/25/2016	Initial publication
2	8/23/2018	Format with SGS brand
3	10/10/2018	• Added `ODOR*` stream types to *Section 2, Stream Concept* • Added `ev_tcp_send`/`ev_tcp_receive` to *Section 5.1, Updating Composition Variables* • Added ODOR Members table to *Section 6.1, Composition Variable Members*
4	1/2/2019	Revised Section 2, Stream Concept to add clarification of *stream* term
5	4/9/2020	Retrofit to new template Reflowed topic structure
6	5/29/2020	Fix typo: `gasf` > `gasfl` in *Section 5.1 Updating Composition Variables* on page 9.
7	9/20/2021	Revisions to remove use of volume flow rate calculations in: • *Section 3.3 Typical go.scp Launch Script* on page 4 • *Section 5.2 Updating Property Variables* on page 10 Added hyperlinked cross-references to usage help on cyflex.com as applicable throughout the document.
8	6/20/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1 Overview* on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

LIST OF TABLES

1 Overview

CyFlex uses two data types to describe to characteristics of fluids and the properties of a point in a fluid stream.

1. The composition variable contains information about the chemical makeup of a fluid.
2. The property variable contains information about fluids properties as a function of temperature and pressure.

Refer to the following for details of the variable contents:

- *Section 6.1 Composition Variable Members* on page 11
- *Section 6.2 Property Variable Members* on page 13

2 Stream Concept

Each composition variable is associated with a fluid stream. A stream may have changing properties at different points in the stream due to changes in temperature or pressure, but the composition does not vary from one point to another within a stream. What we refer to as a "stream" does not have any chemical reactions taking place and does not have any mixing with other components. Multiple streams can be mixed to form a new stream. Refer to *Computing Gas Stream Composition and Gas Flow* for information about mixing streams.

GASOLINE, DIESEL_FUEL, and UREA are assumed to be liquids, while the others are assumed to be gases.

CyFlex supports 29 streams. The streams use one of the following stream identifiers listed the sub-sections below.

Notes:

The *_DP streams were intended for batch processing of conditions for a "datapoint".

The ODOR* stream types are used for Mercaptan odorant measurements.

2.1 Gaseous Streams Identifiers

COMBUSTION_AIR
COMBUSTION_AIR_DP
NATURAL_GAS
GAS1 (user defined)
GAS2 (user defined)
AIR_FUEL_MIX
AIR_FUEL_MIX_DP
EXHAUST
EXHAUST_DP
STACK
STACK_DP
EGR_MIX
EGR_MIX_DP
AFTER-TREATMENT1
AFTER-TREATMENT1_DP
AFTER-TREATMENT2
AFTER-TREATMENT2_DP
MINI_TUNNEL
MINI_TUNNEL_DP
DILUTION_TUNNEL
DILUTION_TUNNEL_DP
GASOLINE_VAPOR

2.2 Liquid Streams Identifiers

GASOLINE
DIESEL_FUEL
UREA

2.3 Odor Streams Identifiers

ODOR1
ODOR2
ODOR3
ODOR4

3 Creating Composition and Property Variables

3.1 Prerequisite

Many of the fluid properties are computed from a CHEMKIN properties database defined by the file `/specs/properties/therm.dat`. This file must exist for property values to be computed. The contents of this file never change and a copy can be obtained from the central node `/cyflex/specs.def/properties/` directory if it is not part of the initial system installation.

The data in this file is read by the `init_properties` program which extracts information needed for the various defined components of composition variable.

3.2 Creating the Variables

Use `init_properties` to create composition and property variables. Typically, `init_properties` is launched from the `go.scp` startup script. This creates the variables and initializes certain attributes of the variables, such as the units, display resolution, and the member extender strings, but does not assign the variable labels. It also computes the molar mass of each of the composition components and initializes some of the properties of those components, such as specific heat and heating value.

Launch `init_properties` from `go.scp` as shown in *Section 3.3 Typical go.scp Launch Script* below.

3.3 Typical go.scp Launch Script

For test systems which perform flow and property computations, the following example shows a typical launch sequence in `/cell/go.scp`.

```
##########################################################################
#
# The section below deals primarily with fluid streams that supply the
# engine with air or fuel.
#
# Various applications are available to perform computations of fluid
# composition, properties, and flow rates.
#
##########################################################################

##########################################################################
#  "init_properties" and "init_composition" must precede the first
#  copy of "gasfl" so that the gas composition variable will be
#  initialized
#
# init_properties creates the memory are for composition and property
# variables - required for "gasfl", "gas_prop", "gas_mix" , "burn_emis"
##########################################################################
```

```
init_properties

   ##########################################################################
   # init_compositon reads /specs/properties/comp_specs.NNN and initializes
   # the values of composition variables to the last value saved when last
   # running or those permanently defined by a comp.<STREAM> file
   ##########################################################################

init_composition

   ##########################################################################
   # update_composition receives the "onga_onga" event to update the
   # composition of natural gas fuel (the NATURAL_GAS stream)
   # use this only in test cells using natural gas fuel or those that have
   # abatement systems with natural gas burners.
   ##########################################################################

update_composition &

   ##########################################################################
   # gas_prop computes the properties of one or more streams
   ##########################################################################

gas_prop 15 FAS /specs/properties/prop_specs.$cell &

   ##########################################################################
   # gasfl is used to compute the mass flow rate based on
   # pressure, and temperature - it also computes the composition of the
   # air/water vapor mixture in combustion air
   ##########################################################################

#gasfl 12 SLO /specs/af_specs_ca.$cell &
#gasfl 12 SLO /specs/af_specs.$cell &
#gasfl 12 SLO /specs/af_specs.pms &

   ##########################################################################
   # subsonic is used to compute the mass flow rate and volume flow rate,
   # based on measurements taken on a subsonic venturi
   ##########################################################################

subsonic 12 MED /specs/subsonic_spec.R &
subsonic 12 MED /specs/subsonic_spec.L &

   ##########################################################################
   # add_water is used in conjunction with gas flow applications such as
   # subsonic, lfe, and critical_flow if the stream contains moisture which
   # can be measured
   ##########################################################################

add_water 11 SLO /specs/properties/addwater_specsL.$cell &
add_water 11 SLO /specs/properties/addwater_specsR.$cell &

   ##########################################################################
   # volef computes the combustion air volumetric efficiency of an engine
   ##########################################################################
```

```
#volef 12   SLO /specs/volef_L.$cell &
#volef 12   SLO /specs/volef_R.$cell &
#volef 12   SLO /specs/volef.$cell &


   ##########################################################################
   # gas_mix computes the composition of more than one stream which are
   # mixed together
   ##########################################################################

gas_mix  12 SLO /specs/properties/mix_specs.$cell &

   ##########################################################################
   # set_composition task allows Cyflex to update a composition variable with
   # real time values.  This is used only when there is a local gas
   # composition analyzer that is capable of updating rapidly.
   ##########################################################################

set_composition 11 SLO /specs/properties/comp.TFS &
```

4 Initializing Composition Variables

Use the `init_composition` process to create composition variables. This program will read a specification file and create the appropriate variables. Each variable must have a stream identifier using one of the types listed in *Section 2 Stream Concept* on page 2, such as NATURAL_GAS, COMBUSTION_AIR, etc. The variable name is arbitrary, except that it must end with a '.' (dot) and by convention always ends with **C.**, for example `inlet_airC.` The variable definition also includes the pathname of a file which is used to initialize the composition variable. Some composition variables may remain at a constant value based on the values in the initialization file, while most will be updated by a specific memory resident application. An example of a variable which would not change might be that for a stream of injected propane. The variable would contain a mole fraction of 1.00 for the propane component and would remain at that value. Most other composition variable values will change with time as they are modified by some process.

The specification file read by the `init_composition` application can be located anywhere, but by convention is always located in the `/specs/properties/` directory and has a name such as `comp_specs.`*NNN*, where *NNN* is the test cell name. The default path for the spec file is always `/specs/properties/comp_specs.`*NNN* but may be optionally specified as any other pathname.

Syntax:

```
init_composition [spec file pathname]
```

where:

`spec_file_pathname`: Optionally specify the pathname to the file that defines all of the composition variables to be created and defines the files which contain the initial values. The default is `/specs/properties/comp_specs.`*NNN*.

Note:

`init_composition` creates the variables, initializes them and then terminates. It may be run from the command line if desired to re-initialize the contents.

The following is a format example of the spec file for "`init_composition`": See `/cyflex/specs.def/comp_specs.def`.

```
------------------------------------ comp_specs.NNN-----------------------
#stream_type      variable_label    initialization_filename
COMBUSTION_AIR    inlet_airC.       /specs/properties/comp.COMBUSTION_AIR
NATURAL_GAS       ngC.              /specs/properties/comp.NATURAL_GAS
AIR_FUEL_MIX      faC.              /specs/properties/comp.AIR_FUEL_MIX
EXHAUST           exhC.             /specs/properties/comp.EXHAUST
ODOR1             mercaptanO.       /specs/properties/comp.ODOR1
-------------------------------------------------------------------------
Format example for an initialization file:

------------------------comp.COMBUSTION_AIR--------------------------------
#stream_type
COMBUSTION_AIR
#last update timestamp (time_t format)
1204500000
#list_of_components     mole_fraction
```

```
02                    .20946
N2                    .78087
AR                    .00934
C2                    .00033
```

5 Updating Composition and Property Variables

5.1 Updating Composition Variables

Composition variables may be modified by the following list of applications:

- `add_water`: Use this application to modify a specified composition variable by adding the moisture content to the variable. A specification file defines the stream (composition variable), the total pressure, and the vapor pressure at the point where the total pressure is measured. It is normally used for air streams and must be used in conjunction with the following applications:
  - `subsonic`; see cyflex.com usage help for `subsonic`
  - `lfe`, see cyflex.com usage help for `lfe`
  - `critical_flow`, see cyflex.com usage help for `critical_flow`

  Do not use `add_water` in conjunction with `gasfl` or `venturi` as those applications perform their own moisture addition computations. Refer to cyflex.com usage help for `add_water`.
- `gasfl, venturi`: These applications update the associated stream composition based on the current moisture content of the stream. Refer to cyflex.com usage help for `gasfl` and `venturi`.
- `update_composition`: Use this application for a system which supports one or more gas composition or odorant monitoring device. It receives a message event which defines the current components of a gas being sampled either by the `ongadata` application or the `set_composition` application. The incoming event name is `onga_onga`. This application also writes the current composition to the file `/specs/properties/comp.<stream_identifier>` so that when a CyFlex system is restarted, the composition can be initialized to that value. However, for the initialization to occur, the filename in the `comp_specs.NNN` file must have the same name as `specs/properties/comp.<stream_identifier>`. Refer to *Section 4 Initializing Composition Variables* on page 7. Refer to cyflex.com usage help for `update_composition` and `set_composition`.
- `set_composition`: Use this application for a system which supports a composition monitoring device that outputs the component concentrations to real variables. The `set_composition` application will read the concentration values from the variables and package them up into a message event named `onga_onga` which is then received and processed by the `update_composition` application. There may be more than one instance of this application running to handle data from more than one analyzer.

  Typically, this application is used for gas analyzers that have a fairly rapid update rate. It also supports odorant variables and will create an output file for loading into a database of odor component values.
- `ev_tcp_send/ev_tcp_receive`: These applications are general purpose applications that can be used to send CyFlex message events from one node to another. The most common use is to transmit composition information from a central node to test cells or other systems that need stream composition that is collected from a single location but are applicable for multiple systems. Typically, the central node collects the

natural gas composition from a main feeder line to a test facility and this information is shared with all test cells that use natural gas. The `ev_tcp_send` application transmits the data through the network. Each test cell must have `ev_tcp_receive` running as well as the `update_composition` app. The central (sending) node uses a specification file that contains a list of all the destination nodes that will receive the information along with the incoming and outgoing event names. For the common use we describe here, the incoming and outgoing event names are both `onga_onga`.

Refer to *Sending Message Events between CyFlex® Systems* on the Cummins Wiki and to cyflex.com usage help for `ev_tcp_send`/`ev_tcp_receive`.

5.2 Updating Property Variables

Use the `gas_prop` application to modify property variables. The application computes the properties of all gaseous streams. Refer to cyflex.com usage help for `gas_prop` syntax and options.

Note:

`gas_prop` is a memory resident application and continually updates the properties at the rate specified

The following is a format example of the spec file for "`gas_prop`":

```
---------------------------prop_specs.NNN-----------------------------------
NOTE: The vapor pressure entry on the first line is no longer used in
      Cyflex.6.3.26 and newer versions. The vapor pressure will be acquired
      From the composition variable

#ambient_pressure       vapor_pressure_at_ambient
barometer               vap_pa

#up to 32 property computations may be listed
#property_variable_label      comp_variable_label     pres        temp
inlet_airP.                   inlet_airC.             mtr0_p      mtr0_t
comp_outP.                    inlet_airC.             cmp_ot_p    cmp_ot_t
ngP.                          ngC.                    mtr1_p      mtr1_t
faP.                          faC.                    mnf_in_p    mnf_in_p
-----------------------------------------------------------------------------
```

6 Composition and Property Variable Members

6.1 Composition Variable Members

Composition variables contain multiple floating-point values that define the chemical components of a fluid stream. The values consist of the mole fraction of each component. The label of a composition variable always ends in a "dot" or period, for example, `ngC.` Access to the different values is obtained through the use of a 2 or 3-character extender. For instance, the mole fraction of oxygen can be obtained by using the label `ngC.02`.

Table 1 through *Table* 3 on page 12 summarize the extenders and their meaning.

Table 1: Gaseous Members Extenders

Extender	*Formula*	*Component*
H2	h2	hydrogen mole fraction
CO	co	carbon dioxide mole fraction
O2	o2	oxygen mole fraction
WA	h2o	water mole fraction
C2	co2	carbon dioxide mole fraction
N2	n2	nitrogen mole fraction
NX	no2	nitrogen dioxide mole fraction
AR	ar	argon mole fraction
ME	ch4	methane mole fraction
EE	c2h6	ethane mole fraction
PR	c3h8	propane mole fraction
NB	n-c4h10	n-butane mole fraction
IB	i-c4h10	iso-butane mole fraction
NP	n-c5h12	n-pentane mole fraction
IP	i-c5h12	iso-pentane mole fraction
HX		hexanes mole fraction
HE		heptanes mole fraction
OC		octanes mole fraction
NN		nonanes mole fraction
HS	h2s	hydrogen sulfide mole fraction

Table 2: ODOR Members Extenders

Extender	Formula	Component
H2S	h2s	Hydrogen sulfide (as odor component)
MTM	CH3-SH	Methyl Mercaptan
ETM	CH3CH2-SH	Ethyl Mercaptan
DMS	CH3-S-CH3	Dimethyl Sulfide
IPM	(CH3)2-CH-SH	Iso 2-Propyl Mercaptan
TBM	(CH3)3-C-SH	tert Butyl Mercaptan
MES	CH3CH2-S-CH3	Methyl Ethyl Sulfide
NPM	CH3CH2CH2-SH	(n) 1-Proply Mercaptan
SBM	CH3CH2CH(CH3)-SH	(sec) 2-Butyl Mercaptan
DES	CH3CH2S-CH2CH3	Diethyl Sulfide
THT	C4H8S	TetraHyrdoThiophene

Table 3: Liquid Members Extenders

Extender	Formula	Component
NM	n	nitrogen liquid mole fraction
OM	o	oxygen liquid mole fraction
HM	h	hydrogen liquid mole fraction
CM	c	carbon liquid mole fraction
NA	n	nitrogen liquid atoms
OA	o	oxygen liquid atoms
HA	h	hydrogen liquid atoms
CA	c	carbon liquid atoms
NF		nitrogen liquid weight fraction
OF		oxygen liquid weight fraction
HF		hydrogen liquid weight fraction
CF		carbon liquid weight fraction
FA		aromatics fraction
OL		olefins fraction

Extender	Formula	Component
SA		saturates fraction
UT		last update time
MM		stream molar mass
NM	n	nitrogen liquid mole fraction

6.2 Property Variable Members

Property variables contain multiple floating-point values that define the properties of a fluid stream at a particular pressure and temperature.

The label of a property variable always ends in a "dot" or period, for example, ngP, Access to the different values is obtained through the use of a 2-character extender. For instance, the absolute temperature can be obtained by using the label "ngP.AT".

Table 4 summarizes the extenders and their meaning.

Table 4: Property Variable Extenders

Extender	Units	Property
AT	deg_R	temperature
AP	in_hg	pressure
CP	btu/lb/deg_R	specific heat at constant pressure
CV	btu/lb/deg_R	specific heat at constant volume
GM	none	ratio of specific heats
TH	btu/lb	enthalpy
GC	btu/lb/deg_R	gas constant
ER	none	equivalence ratio
SR	none	stoichiometric fuel/air ratio
HV	btu/lb	heating value
HU	%	humidity
ET	BTU	entropy
VS	lb/(ft-sec)	viscosity
RE	none	real relative density
DD	lb/ft3	real gas density
ID	lb/ft3	ideal gas density
IR	none	ideal relative gas density
CM	none	compressibility

Extender	Units	Property
DW	deg_f	dewpoint temperature
AH	lbm	absolute humidity
VP	in_hg	water vapor pressure (in air)
WD	none	wet to dry mole ratio
EA	none	excess air ratio
LM	btu/lb	lower heating value by mass
LV	-	lower heating value by volume
HM	btu/lb	higher heating value by mass
HH	-	higher heating value by volume
WL	-	Wobbe lower heating value
WH	-	Wobbe higher heating value
MR	none	methane number
MO	none	motor octane number
KE	none	knock line exponent
KT	none	knock line temperature coefficient
KP	none	knock line pressure coefficient
HC	none	hydrogen to carbon mole ratio
RD	none	stoichiometric air/fuel ratio dry
RW	none	stoichiometric air/fuel ratio wet
IE	btu/lb	internal energy
OT	none	octane number
CU	none	cetane number
CI	none	cetane index
DS	none	distillation weight as a function of temperature
LH	btu/lb	lower heating value
DE	lb/ft3	density as a function of temperature
UT	sec	oldest composition update time

7 Applications using Composition and Variable Properties

The following applications rely on the existence of composition or property variables and have specification files which reference the variables.

- `gasfl`
- `subsonic`
- `lfe`
- `critical_flow`
- `cfv_1065`
- `Vcone_flow`
- `volef`
- `volef2 (2-cycle engines)`
- `gas_mix`
- `gas_blend`
- `burn_emis`
- `gas_prop`
- `set_composition`
- `init_composition`
- `update_composition`