

CHEMCAD 5 EXCEL UNIT

User Manual

Chemstations, Inc.

INTRODUCTION

A new unit operation, "Excel Unit", has been added to CHEMCAD 5. With the addition of this unit, CHEMCAD 5 is now capable of linking its flowsheet with Excel workbooks as its unit operations. When placing an Excel Unit icon on a flowsheet, specifying the path to an Excel workbook, and running the flowsheet, CHEMCAD can activate the specified Excel workbook and execute the given VBA macros of the workbook. Furthermore, CHEMCAD exports a set of COM interfaces to provide an adequate modeling environment to users of Excel Unit. Every time a VBA macro is executed, a handle to a COM object in CHEMCAD 5 is passed in as an argument. Through the object handle, user can access a number of CHEMCAD functions within their macros. By developing user macros, a list of tasks can be accomplished. User can retrieve the topology of the current flowsheet, such as stream and unit IDs, and their connections. From these IDs, user can retrieve and update stream data and unit specifications. For stream data, user may also apply flash calculations, K-value and enthalpy calculations. Engineering unit conversions are made available for stream data, unit specifications, and general unit conversion cases.

The Excel Unit in CHEMCAD is developed using Microsoft COM with Excel API interfaces. The result is a versatile, user-friendly, powerful and expandable environment for CHEMCAD 5 users. Excel Unit can be designed for a variety of applications. Among these are modeling of user-specific unit operations, making presentation or documentation of CHEMCAD calculation results in Excel, passing user stream data or equipment specifications into CHEMCAD, and monitoring or regulating the calculation states of stream or units on the flowsheet. The user-specific tasks that can only be accomplished using Calculators or User-Added Modules can now be solved within Excel Units. Since VBA is an interpretive language, it does save the trouble of compiling user codes and linking with CHEMCAD libraries.

The steps of completing an Excel Unit are simple to follow. First, place an icon of Excel Unit on the current flowsheet, and make stream connections to or from other unit icons, feeds or products. Second, switch to simulation mode, double click the Excel Unit icon to open the build-in Excel Unit dialog box, and examine the unit dialog files, the Excel workbook path and the names of the macros to be executed during a simulation run. Third, program VBA macros in the specified Excel workbook. Finally, run simulations from CHEMCAD 5, debug through the Excel macros.

Four files are required to support an Excel Unit. They are screen layout (*.my), variable map (*.map), labels for report (*.lab), and an Excel workbook (*.xls). As default, these four files are created under the current job directory with current case name followed by respective extensions. For example, the current job directory is "CC5Data\MyExcelJob" and a case "CaseA" is loaded. When user double-clicks the Excel Unit icon the first time, four files CaseA.my, CaseA.map, CaseA.lab and CaseA.xls are created under directory "CC5Data\MyExcelJob". These files can be renamed or modified to meet user's needs. Dialog layout file, map file and label file need to have identical name. These files are necessary for entering parameters of the Excel Unit. The parameters can later be retrieved within the Excel macros during simulation runs. User can find more information on creating unit specification dialog files elsewhere [1]. The default version of Excel workbook copies the first inlet to the first outlet for its Excel Unit. The user can change Excel workbook path, and should modify or provide macros for their intended calculations.

Many CHEMCAD 5 functions are made available in VBA macros through a number of exported interfaces. This document presents a detailed discussion on the exported interfaces for use with Excel Units. Users of Excel Unit will request the services of one or more interfaces in their VBA macros. The readers should be familiar with VBA syntax and some knowledge of Microsoft COM.

STARTING POINT

Starting from this section, we will discuss CHEMCAD 5 exported interfaces for Excel macros. The first interface is one that will lead to all other interfaces. The only role of this interface is to allow user to reach other interfaces. Only after getting hold of other interfaces, can a user use any CHEMCAD 5 calculations. Hence we start with the description of this interface - ICHEMCADEntry.

Description ICHEMCADEntry interface is the entry point to all other interfaces. When calling an Excel workbook macro during a simulation run, CHEMCAD 5 always passes a handle of this interface to the VBA macro as shown in the following example.

```
Sub MyMacro (ByVal ChemCADEntry As Object)
On Error Resume Next

' Get ChemCAD objects
Dim curUnitOp As Object
Dim strInfo As Object
Dim uopInfo As Object

Set curUnitOp = ChemCADEntry.GetCurUnitOp
Set strInfo = ChemCADEntry.GetStreamInfo
Set uopInfo = ChemCADEntry.GetUnitOpInfo

...

End Sub
```

The writer of a macro should determine the selection of other interfaces to bring into his/her subroutine to meet the calculation needs. In this example, three interfaces are obtained through the methods of ICHEMCADEntry. They are ICurUnitOp, IStreamInfo and IUnitOpInfo. With the handles of these interfaces, a user can access the methods provided by these interfaces. The functionality of these interfaces and those of others will be explained in detail in the following sections.

Definition:

```
dispinterface ICHEMCADEntry
{
methods:
    VARIANT GetFlash();
    VARIANT GetStreamUnitConversion();
    VARIANT GetCurUnitOp();
    VARIANT GetEnthalpy();
    VARIANT GetKValues();
    VARIANT GetFlowsheet();
    VARIANT GetStreamInfo();
    VARIANT GetStreamProperty();
    VARIANT GetUnitOpInfo();
    VARIANT GetUnitOpSpecUnitConversion();
    VARIANT GetEngUnitConversion();
    VARIANT GetCompPPData();
    boolean GetSaveWorkBookFlag();
    boolean SetSaveWorkBookFlag(boolean bSaveAfterRun);
};
```

Methods

Each of the methods exposed by ICHEMCADEntry takes a void argument and returns an IDispatch pointer of another interface to the calling subroutine.

<i>Method</i>	GetFlash
<i>Returns</i>	a handle to IFlash
<i>Method</i>	GetStreamUnitConversion
<i>Returns</i>	a handle to IStreamUnitConversion
<i>Method</i>	GetCurUnitOp
<i>Returns</i>	a handle to ICurUnitOp
<i>Method</i>	GetEnthalpy
<i>Returns</i>	a handle to IEnthalpy
<i>Method</i>	GetKValues
<i>Returns</i>	a handle to IKValues
<i>Method</i>	GetFlowsheet
<i>Returns</i>	a handle to IFlowsheet
<i>Method</i>	GetStreamInfo
<i>Returns</i>	a handle to IStreamInfo
<i>Method</i>	GetStreamProperty
<i>Returns</i>	a handle to IStreamProperty
<i>Method</i>	GetUnitOpInfo
<i>Returns</i>	a handle to IUnitOpInfo
<i>Method</i>	GetUnitOpSpecUnitConversion
<i>Returns</i>	a handle to IUnitOpSpecUnitConversion
<i>Method</i>	GetEngUnitConversion
<i>Returns</i>	a handle to IEngUnitConversion
<i>Method</i>	GetCompPPData
<i>Returns</i>	a handle to ICompPPData

The following two methods give Excel UnitOp user the control to decide when to save the current workbook. By default CHEMCAD saves the workbook after every run of the current Excel UnitOp. Alternatively, user can set the flag to FALSE during a run using SetSaveWorkBookFlag to skip saving workbook after the run.

<i>Method</i>	GetSaveWorkBookFlag
<i>Returns</i>	a boolean flag
<i>Method</i>	SetSaveWorkBookFlag
<i>Returns</i>	a boolean flag

Argument	Type	Description
bSaveAfterRun	boolean	CHEMCAD saves workbook if set TRUE

FLWSHEET

This section describes the interfaces from which the topology of current flowsheet can be established. Two interfaces, ICurUnitOp and IFlowsheet, are discussed.

Description ICurUnitOp interface gives the counts and IDs of the inlets and outlets of the current Excel Unit. User can also get the unit ID of the current Excel Unit, loop count and maximum number of runs.

Definition

```
dispinterface ICurUnitOp
{
methods:
    short GetNoOfInlets();
    short GetInletIDs(VARIANT inletIDs);
    short GetNoOfOutlets();
    short GetOutletIDs(VARIANT outletIDs);
    short GetCurUnitOpID();
    short GetLoopCount();
    short GetLoopLimit();
    short GetRerunFlowsheetFlag();
    short SetRerunFlowsheetFlag(short rerunIfOne);
};
```

Methods

The following two methods give the count and the IDs of inlets. The ID array need to be defined in VBA macro before calling GetInletIDs with lower bound at one and upper bound at no less than the inlet count.

- Method* GetNoOfInlet
- Returns* count of inlets

- Method* GetInletIDs
- Returns* count of inlet IDs returned in array inletIDs

Argument	Type	Description
inletIDs	short[]	Integer array containing inlet IDs [out]

The following two methods give the count and the IDs of outlets. The ID array need to be defined in VBA macro before calling GetOutletIDs with lower bound at one and upper bound at no less than the outlet count.

- Method* GetNoOfOutlets
- Returns* count of outlets

- Method* GetOutletIDs(VARIANT outletIDs)
- Returns* count of outlet IDs returned in array outletIDs

Argument	Type	Description
outletIDs	short[]	Integer array containing outlet IDs [out]

The next group of methods gives the ID of current Excel Unit, current loop count and the maximum number of runs respectively.

Method GetCurUnitOpID
Returns ID of the current Excel Unit

Method GetLoopCount
Returns current loop number

Method GetLoopLimit
Returns maximum number of runs

A significant run time variable RerunFlag allow user to control flowsheeting simulations from Excel macros. Setting the flag to be one forces CHEMCAD 5 to repeat the simulations even the flowsheet has converged. User can stop repeated simulations by resetting the flag to zero.

Method GetRerunFlowsheetFlag / SetRerunFlowsheetFlag
Returns current value of RerunFlag

Description IFlowsheet provides methods for querying the connections of streams and unit operations on the current flowsheet. For a given stream, the source and target unit can be determined. If a unit ID is known, the inlets and outlets of the unit can be queried. All feed streams or product streams can be identified.

Definition

```
dispinterface IFlowsheet
{
methods:
    short GetNoOfStreams();
    short GetNoOfUnitOps();
    short GetAllStreamIDs(VARIANT idArray);
    short GetAllUnitOpIDs(VARIANT idArray);
    short GetStreamCountsToUnitOp(short unitOpID, short* nInlets,
short* nOutlets);
    short GetStreamIDsToUnitOp(short unitOpID, VARIANT idArray);
    short GetInletStreamIDsToUnitOp(short unitOpID, VARIANT idInlets);
    short GetOutletStreamIDsToUnitOp(short unitOpID, VARIANT
idOutlets);
    short GetSourceAndTargetForStream(short streamID, short* sourceID,
short* targetID);
    short GetNoOfFeedStreams();
    short GetNoOfProductStreams();
    short GetFeedStreamIDs(VARIANT idArray);
    short GetProductStreamIDs(VARIANT idArray);
};
```

Methods Stream IDs can be retrieved for the entire flowsheet, streams around a unit, feed stream only or product stream only using

Method GetNoOfStreams
Returns count of all streams on the current flowsheet

Method GetAllStreamIDs
Returns count of stream IDs returned in idArray

Argument	Type	Description
idArray	short[]	Integer array containing all stream IDs [out]

Method GetStreamCountsToUnitOp
Returns total number of streams connected to the given unit

Argument	Type	Description
unitOpID	short	ID of a unit on the flowsheet [in]
nInlets	short	count of inlets to the unit [out]
nOutlets	short	count of outlets to the unit [out]

Method GetStreamIDsToUnitOp / GetInletStreamIDsToUnitOp / GetOutletStreamIDsToUnitOp
Returns total number of IDs returned in idArray, idInlets or idOutlets

Argument	Type	Description
unitOpID	short	ID of a unit on the flowsheet [in]
idArray	short[]	IDs of inlets (>0) and outlet (<0) to the unit [out]
idInlets	short[]	IDs of inlets to the unit [out]
idOutlets	short[]	IDs of outlets to the unit [out]

Method GetNoOfFeedStreams
Returns count of feed streams on the current flowsheet

Method GetFeedStreamIDs
Returns count of feed stream IDs returned in idArray

Argument	Type	Description
idArray	short[]	Integer array containing feed stream IDs [out]

Method GetNoOfProductStreams
Returns count of product streams on the current flowsheet

Method GetProductStreamIDs
Returns count of product stream IDs returned in idArray

Argument	Type	Description
idArray	short[]	Integer array containing product stream IDs [out]

Unit IDs can be retrieved for the entire flowsheet or a given stream using

Method GetNoOfUnitOps
Returns count of unit operations on the current flowsheet

Method GetAllUnitOpIDs
Returns count of unit IDs returned in idArray

Argument	Type	Description
idArray	short[]	Integer array containing all unit IDs

Method GetSourceAndTargetForStream
Returns number of unit IDs returned

Argument	Type	Description
streamID	short	ID of a stream on the flowsheet [in]
sourceID	short	count of inlets to the unit [out]
targetID	short	count of outlets to the unit [out]

Example

The example shows how to make a table of the inlets and outlets for all the units on the current flowsheet. In this example, the inlets appear as positive integers, and the outlets as negative integers.

```

Sub PrintFlowsheetIDs(ByVal ChemCADEntry As Object)
On Error Resume Next

' get cc5 object

Dim flowsheet As Object
Set flowsheet = ChemCADEntry.GetFlowsheet

'streams and unitops
Dim streamCount As Integer
Dim unitOpCount As Integer
streamCount = flowsheet.GetNoOfStreams
unitOpCount = flowsheet.GetNoOfUnitOps

Dim check As Integer
Dim streamIDs(1 To 100) As Integer
Dim unitOpIDs(1 To 100) As Integer

check = flowsheet.GetAllStreamIDs(streamIDs)
If (check <> streamCount) Then
    MsgBox "Not all stream IDs are returned"
End If

check = 0
check = flowsheet.GetAllUnitOpIDs(unitOpIDs)
If (check <> unitOpCount) Then
    MsgBox "Not all unitOp IDs are returned"
End If

'feed and product streams
Dim feedCount, prodCount As Integer
Dim feedIDs(1 To 100) As Integer
Dim prodIDs(1 To 100) As Integer

feedCount = flowsheet.GetNoOfFeedStreams
check = flowsheet.GetFeedStreamIDs(feedIDs)
If (check <> feedCount) Then
    MsgBox "Not all feed stream IDs are returned"
End If

prodCount = flowsheet.GetNoOfProductStreams
check = flowsheet.GetProductStreamIDs(prodIDs)
If (check <> prodCount) Then
    MsgBox "Not all product stream IDs are returned"
End If

Dim curUnitIndex As Integer
Dim curID As Integer
Dim curInletIndex As Integer
Dim curOutletIndex As Integer
Dim curStreamIndex As Integer
Dim curRow As Integer
Dim curCol As Integer
Dim id As Integer
Dim dummy As Integer

Dim flowSht As Worksheet
Set flowSht = Worksheets("FLOWSHEET")

```

```

flowSht.Activate

curRow = 1
curCol = 1
flowSht.Cells(curRow, curCol).value = "UnitOp ID"
curCol = 3
flowSht.Cells(curRow, curCol).value = "Stream ID"

curRow = 2
For curUnitIndex = 1 To unitOpCount Step 1
    Dim inCount As Integer
    Dim outCount As Integer
    Dim idArray(1 To SIZE_STREAM_ARRAY * 2) As Integer

    curID = unitOpIDs(curUnitIndex)
    check = -1
    check = flowsheet.GetStreamCountsToUnitOp(curID, inCount, outCount)

    check = -1
    check = flowsheet.GetStreamIDsToUnitOp(curID, idArray)

    curRow = curRow + 1
    flowSht.Cells(curRow, 1).value = curID

    curCol = 2
    For curStreamIndex = 1 To (inCount + outCount) Step 1
        curCol = curCol + 1
        id = idArray(curStreamIndex)
        flowSht.Cells(curRow, curCol).value = id
    Next curStreamIndex
Next curUnitIndex

End Sub

```

STREAM DATA

Stream data define the thermodynamic state and the rates of transportation of processing materials on a flowsheet. The data may be used in making operational decisions or satisfying computational needs. CHEMCAD 5 allows access of the data from Excel macros through interface IStreamInfo and IStreamProperty.

Description With a known stream ID, IStreamInfo allows to retrieve and update stream data, including stream label, temperature, pressure, mole fraction, enthalpy, and component flowrates. All the engineering quantities are given in CHEMCAD 5 internal units. The component names and their IDs are also provided through the methods on this interface.

Definition

```

dispinterface IStreamInfo
{
methods:
    short GetNoOfComponents();
    short PutStreamByID(short streamID, float tempR, float presPsia,
float moleVapFrac, float enthBtu_Hr, VARIANT compFlowLbmol_Hr);

    short GetStreamByID(short streamID, float* tempR, float* presPsia,
float* moleVapFrac, float* enthBtu_Hr, VARIANT compFlowLbmol_Hr);

    short GetIDsOfComponents(VARIANT compIDs);
    BSTR GetComponentNameByPosBaseOne(short compPos);
    short GetComponentIDByPosBaseOne(short compPos);
    BSTR GetStreamLabelByID(short streamID);
};

```

Methods

Two key method, GetStreamByID and PutStreamByID, are for retrieving and updating stream data.

Method GetStreamByID
Returns count of returned component data items in compFlowLbmol_Hr

Argument	Type	Description
streamID	short	ID of a stream [in]
tempR	float	Temperature (R) [out]
presPsia	float	Pressure (psia) [out]
moleVapFrac	float	Mole vapor fraction [out]
enthBtu_Hr	float	Enthalpy (Btu/hr) [out]
compFlowLbmol_Hr	float[]	Component flowrates (lbmol/hr) [out]

Method PutStreamByID
Returns count of received data items from compFlowLbmol_Hr

Argument	Type	Description
streamID	short	ID of a stream [in]
tempR	float	Temperature (R) [in]
presPsia	float	Pressure (psia) [in]
moleVapFrac	float	Mole vapor fraction [in]
enthBtu_Hr	float	Enthalpy (Btu/hr) [in]
compFlowLbmol_Hr	float[]	Component flowrates (lbmol/hr) [in]

Other methods on the interface are for convenience.

Method GetStreamLabelByID
Returns user label string of the stream

Argument	Type	Description
streamID	short[]	stream ID for which the label is retrieved [in]

Method GetIDsOfComponents
Returns count of component IDs returned in compIDs

Argument	Type	Description
compIDs	short[]	component IDs

Method GetComponentIDByPosBaseOne(short compPos);
Returns ID of the component at position compPos on the component list

Method GetComponentNameByPosBaseOne (short compPos);
Returns name string of the component

Argument	Type	Description
compPos	short[]	component position on the component list [in]

Description For a user-supplied stream, all stream properties can be retrieved from IStreamProperty interface. Each item of stream properties is identified by an integer as given on the tables below.

Property ID	Stream Property Description (components)	Units
-i	Mole flow rate of the ith component	Lbmol/hr
-(i+200)	Mass flow rate of the ith component	lb/hr
-(i+400)	Std liquid volume flow rate of the ith component	ft3/hr
-(i+600)	Mole fraction of the ith component	
-(i+800)	Mass fraction of the ith component	
-(i+1000)	Std liquid volume fraction of the ith component	

Property ID	Stream Property Description (total)	Units
1	Temperature	R
2	Pressure	psia
3	Mole vapor fraction	
4	Enthalpy	Btu/hr
5	Total mole rate	lbmol/hr
6	Total mass rate	lb/hr
7	Total std liquid volume rate	ft3/hr
8	Total std vapor volume rate	ft3/hr
9	Total actual volume rate	ft3/hr
10	Total actual density	lb/ft3
11	Total Mw	
12	Gross H value	
13	Net H value	
14	Reid vapor pressure	
15	UOPK	
16	VABP	
17	MeABP	
18	Flash point	
19	Pour point	
20	Total entropy	
21	Mass vapor fraction	
22	PH value	

Property ID	Stream Property Description (vapor)	Units
26	Vapor mole rate	
27	Vapor mass rate	
28	Vapor enthalpy	
29	Vapor entropy	
30	Vapor Mw	
31	Vapor actual density	
32	Vapor actual volume rate	
33	Vap std. Liquid volume rate	
34	Vap std. Vapor volume rate	
35	Vapor cp	
36	Vapor Z factor	
37	Vapor viscosity	
38	Vapor thermal conductivity	

Property ID	Stream Property Description (liquid)	Units
41	Liquid mole rate	
42	Liquid mass rate	
43	Liquid Enthalpy	
44	Liquid Entropy	
45	Liquid Mw	
46	Liquid actual density	
47	Liquid actual volume rate	
48	Liquid std. Liquid volume rate	
49	Liquid std. Vapor volume rate	
50	Liquid cp	
51	Liquid Z factor	
52	Liquid viscosity	
53	Liquid thermal conductivity	
54	Liquid surface tension	

Definition

```

dispinterface IStreamProperty
{
methods:

    short GetNoOfComponents();
    short DefineStream(float tempR, float presPsia, float moleVapFrac,
float   enthBut_Hr, VARIANT compFlowLbmol_Hr);

    float GetStreamProperty(short propID);
};

```

Methods

To calculate stream properties, user first has to define a stream using

Method DefineStream
Returns number of data items received from array compFlowLbmol_Hr

Argument	Type	Description
tempR	float	Temperature (R) [in]
presPsia	float	Pressure (psia) [in]
moleVapFrac	float	Mole vapor fraction [in]
enthBtu_Hr	float	Enthalpy (Btu/hr) [in]
compFlowLbmol_Hr	float[]	Component flowrates (lbmol/hr) [in]

After a valid stream is specified, its properties can be retrieved one at a time from

Method GetStreamProperty
Returns stream property

Argument	Type	Description
propID	short	Stream property ID [in]

Example

UNIT SPECIFICATION DATA

Each unit operation on a flowsheet is specified by assigning values to a set of parameters. This set of parameters are used during the simulations to determine the interior processing conditions of the unit. Some parameters may need to be updated during simulation runs.

Description IUnitOpInfo interface allows user to retrieve and update unit specification parameters. It also answers the query for user-supplied unit label string.

Definition

```

dispinterface IUnitOpInfo
{
  methods:
    short GetUnitOpSpecArrayDimension();
    short GetUnitOpSpecByID(short unitOpID, VARIANT unitOpSpec);
    short PutUnitOpSpecByID(short unitOpID, VARIANT unitOpSpec);
    BSTR GetUnitOpLabelByID(short unitOpID);
    short GetUnitOpCategoryByID(short unitOpID, BSTR* unitOpCate);
};

```

Methods Each unit can store up to 250 parameters in an array of float type. With a known unit ID, user can retrieve or update unit specification parameters.

Method GetUnitOpSpecByID
Returns number of parameters returned in unitOpSpec

Argument	Type	Description
unitOpID	short	Unit ID [in]
unitOpSpec	float[]	Unit specification parameters [out]

Method PutUnitOpSpecByID
Returns number of parameters received from unitOpSpec

Argument	Type	Description
unitOpID	short	Unit ID [in]
unitOpSpec	float[]	Unit specification parameters [in]

Method GetUnitOpLabelByID
Returns user-defined unit label

Argument	Type	Description
unitOpID	short	Unit ID [in]

Each unit has its own identity. This identify informs us the role of the unit on a flowsheet, such as mixer, flash, or kinetic reactor. In CHEMCAD 5, this identify is given by either a unique category ID or a four-character string.

Method GetUnitOpCategoryByID
Returns unit category ID

Argument	Type	Description
UnitOpID	short	Unit ID [in]
unitOpCate	BSTR	4 character string of category name

Example

FLASH

Description IFlash interface provides access to CHEMCAD 5 flash calculations. To use flash calculations, a user first defines a feed stream, including its temperature, pressure, enthalpy and its component flowrates. All quantities should be given in CHEMCAD 5 internal units. For a user-defined feed stream, TP (isothermal flash at given temperature and pressure), VP (flash at given mole vapor fraction and pressure), VT (flash at given mole vapor fraction and temperature), HP (adiabatic flash at given pressure) flash calculations are available. After a flash calculation is converged, the calculated liquid and vapor stream can be retrieved. The k-values at equilibrium and ion flow rates (if electrolyte is chosen) may also be returned.

Definition

```
dispinterface IFlash
{
methods:
    long GetNoOfComponents();
    short DefineFeedStream(float tempR, float presPsia, float
enthBtu_Hr, VARIANT compFlowLbmol_Hr);

    short CalculateHPFlash(float enthBtu_Hr, float presPsia);
    short GetVaporStream(float* tempR, float* presPsia, float*
enthBtu_Hr, float* rateLbmol_Hr, VARIANT compFlowLbmol_Hr);

    float GetMoleVaporFraction();
    float GetCalculatedHeatDuty();
    short CalculateTPFlash(float tempR, float presPsia);
    short CalculateVPFlash(float moleVapFrac, float presPsia);
    short CalculateVTFlash(float moleVapFrac, float tempR);
    short GetKValues(VARIANT kValues);
    short GetIonRates(VARIANT ionFlowLbmol_Hr);
    short GetLiquidStream(float* tempR, float* presPsia, float*
enthBtu_Hr, float* rateLbmol_Hr, VARIANT compFlowLbmol_Hr);
};
```

Methods

The methods defined in this interface are devised around flash calculations. The correct calling sequence should be that 1) define feed streams; 2) execute a proper flash calculation; 3) retrieve calculation results. The user can redefine feed stream if multiple streams need to be flashed. For a defined feed stream, the user can also call different flash calculations. After each flash calculation is converged, the calculated data (liquid and vapor stream, etc.) can only be retrieved once.

Before calling the methods, user need to declare component array in VBA macro. The lower bound of a component array has to be one, and the upper bound should be no less than the actual number of components. All the units of the arguments on the methods of this interface are CHEMCAD internal units.

Method GetNoOfComponents
Returns count of components in current job

Method DefineFeedStream
Returns number of flowrates received from compFlowLbmol_Hr

Argument	Type	Description
tempR	float	Temperature (R) [in]
presPsia	float	Pressure (psia) [in]
enthBtu_Hr	float	Enthalpy (Btu/hr) [in]
compFlowLbmol_Hr	float[]	Component flowrates (lbmol/hr) [in]

HP flash at given enthalpy and pressure is carried out through the following method.

Method CalculateHPFlash
Return convergence of the flash routine (0 – converge, 1 – diverge)

Argument	Type	Description
presPsia	float	Pressure (psia) [in]
enthBtu_Hr	float	Enthalpy (Btu/hr) [in]

TP flash at specified temperature and pressure is carried out by the next method.

Method CalculateTPFlash
Return convergence of the flash routine (0 – converge, 1 – diverge)

Argument	Type	Description
tempR	float	Temperature (R) [in]
presPsia	float	Pressure (psia) [in]

VP flash at specified mole vapor fraction and pressure is called through CalculateVPFlash. This method can also calculate bubble point temperature if mole vapor fraction is 0, and dew point temperature if mole vapor fraction if 1.

Method CalculateVPFlash
Returns convergence of the flash routine (0 – converge, 1 – diverge)

Argument	Type	Description
presPsia	float	Pressure (psia) [in]
moleVapFrac	float	Mole vapor fraction [in]

VT flash at specified mole vapor fraction and temperature is called in CalculateVTFlash. This method can also calculate bubble point pressure if mole vapor fraction is 0, and dew point pressure if mole vapor fraction is 1.

Method CalculateVTFlash
Returns convergence of the flash routine (0 – converge, 1 – diverge)

Argument	Type	Description
tempR	float	Temperature (R) [in]
moleVapFrac	float	Mole vapor fraction [in]

After a flash calculation has converged, the following methods can be called to retrieve calculation data.

Method GetVaporStream / GetLiquidStream
Returns number of data items returned in array compFlowLbmol_Hr

Argument	Type	Description
tempR	float	Temperature (R) [out]
presPsia	float	Pressure (psia) [out]
enthBtu_Hr	float	Enthalpy (Btu/hr) [out]
rateLbmol_Hr	float	Total mole rate (lbmol/hr) [out]
compFlowLbmol_Hr	float[]	Component flowrates (lbmol/hr) [out]

Method GetKValues
Returns number of data items returned in array kValues

Argument	Type	Description
kValues	float[]	Calculated k-values [out]

Method GetIonRates
Returns number of data items returned in array ionFlowLbmol_Hr

Argument	Type	Description
ionFlowLbmol_Hr	float[]	Ion rates in liquid phase [out]

Method GetMoleVaporFraction
Returns calculated mole vapor fraction

Method GetCalculatedHeatDuty
Returns calculated heat duty

Example This subroutine shows the calling sequence of methods from IFlash interface

```
Sub MixerInExcel(ByVal ChemCADEntry As Object)
On Error Resume Next

' get all cc5 objects
Dim curUnitOp As Object
Dim strInfo As Object
Dim uopInfo As Object
Dim flash As Object
```

```

Set curUnitOp = ChemCADEntry.GetCurUnitOp
Set strInfo = ChemCADEntry.GetStreamInfo
Set uopInfo = ChemCADEntry.GetUnitOpInfo
Set flash = ChemCADEntry.GetFlash

...

Dim check As Integer
Dim component(1 To SIZE_COMP_ARRAY) As Single
Dim compSum(1 To SIZE_COMP_ARRAY) As Single

...

' HP flash to determine vapor fraction and temperature
check = flash.DefineFeedStream(setTemperature, setPressure, setEnthalpy,
compSum)
check = flash.CalculateHPFlash(setEnthalpy, setPressure)
check = flash.GetVaporStream(temperature, pressure, enthalpy, flowRate,
component)

mvf = flash.GetMoleVaporFraction
heatDuty = flash.GetCalculatedHeatDuty

...

End Sub

```

ENTHALPY

Description IEnthalpy calculates the liquid or vapor enthalpy of a user-supplied stream.

Definition

```

dispinterface IEnthalpy
{
methods:
    short GetNoOfComponents();
    short DefingStream(float tempR, float presPsia, VARIANT
compFlowLbmol_Hr);

    short CalculateLiquidEnthalpy(float* enthBtu_Hr);
    short CalculateVaporEnthalpy(float* enthBtu_Hr);
};

```

Methods

When stream enthalpy is needed, the user call the following function to specify the stream data.

Method DefingStream
Returns number of data items received from compFlowLbmol_Hr

Argument	Type	Description
tempR	float	temperature [in]
presPsia	float	pressure [in]
compFlowLbmol_Hr	float[]	liquid/vapor component flowrates [in]

After the stream data is specified, the enthalpy of the stream can be calculated using one of the functions

Method CalculateLiquidEnthalpy / CalculateVaporEnthalpy
Returns 1 – return value is valid; 0 – the return value is invalid

Argument	Type	Description
enthBtu_Hr	float	enthalpy [out]

Example

K-VALUES

Description IKValues calculates k-values at given temperature, pressure and liquid-vapor compositions. It may also return ion rates if electrolyte is chosen.

Definition

```

dispinterface IKValues
{
methods:
    short GetNoOfComponents();
    short DefineLiquidStream(float tempR, float presPsia, VARIANT
    compFlowLbmol_Hr);

    short DefineVaporStreamComponentRates(VARIANT compFlowLbmol_Hr);

    short GetKValues(VARIANT kValues);
    short GetIonRates(VARIANT ionFlowLbmol_Hr);
    short GetActivityCoefficients(VARIANT actCoef);
    short GetFugacityCoefficients(VARIANT fugCoef);
};

```

Methods The first step is to define the liquid-vapor two phase system, its temperature, pressure and compositions using the two methods given below

Method DefineLiquidStream / DefineVaporStreamComponentRates
Returns number of data items received from array compFlowLbmol_Hr

Argument	Type	Description
tempR	float	temperature [in]
presPsia	float	pressure [in]
compFlowLbmol_Hr	float[]	liquid/vapor component flowrates [in]

The calling sequence of the above methods are arbitrary. As soon as liquid-vapor data are defined, the k-values are calculated. User can call the following methods to get the results

Method GetKValues / GetIonRates
Returns number of data items returned in the argument array

Argument	Type	Description
kValues	float[]	calculated k-values [out]
ionFlowLbmol_Hr	float[]	ion rates in liquid phase [out]

If the activity coefficients or the fugacity coefficients are required, the next two functions can be used

Method GetActivityCoefficients / GetFugacityCoefficients
Returns number of data items returned in the argument array

Argument	Type	Description
actCoef	float[]	Activity coefficients [out]
fugCoef	float[]	Fugacity coefficients [out]

Example

Here we show the use of KValues interface.

```

Sub KValues(ByVal ChemCADEntry As Object)
On Error Resume Next

Dim curUnitOp As Object
Dim strInfo As Object
Dim kValues As Object

Set curUnitOp = ChemCADEntry.GetCurUnitOp
Set strInfo = ChemCADEntry.GetStreamInfo
Set kValues = ChemCADEntry.GetKValues

Dim check As Integer
' inlets
Dim nInlets As Integer
nInlets = curUnitOp.GetNoOfInlets
Dim inletIDs(1 To SIZE_STREAM_ARRAY) As Integer
check = curUnitOp.GetInletIDs(inletIDs)

' outlets
Dim nOutlets As Integer
nOutlets = curUnitOp.GetNoOfOutlets
Dim outletIDs(1 To SIZE_STREAM_ARRAY) As Integer
check = curUnitOp.GetOutletIDs(outletIDs)

Dim nStream As Integer
Dim iStream As Integer
If nInlets > nOutlets Then
    nStream = nOutlets
Else
    nStream = nInlets
End If

' first simply pass the inlet to outlet
Dim temperature As Single
Dim pressure As Single
Dim mvf As Single
Dim enthalpy As Single
Dim component(1 To SIZE_COMP_ARRAY) As Single

Dim vaporstream As Integer
Dim liqstream As Integer

vaporstream = inletIDs(1)
liqstream = inletIDs(2)

Dim result(1 To SIZE_COMP_ARRAY) As Single
Dim dummy As Single
Dim iComp As Integer
Dim nComp As Integer
If vaporstream > 0 And liqstream > 0 Then

    Dim liqID As Integer
    Dim vapID As Integer
    liqID = liqstream
    vapID = vaporstream

    check = 0
    check = strInfo.GetStreamByID(liqID, temperature, pressure, mvf,
    enthalpy, component)
    check = kValues.DefineLiquidStream(temperature, pressure, component)

```

```

check = 0
check = strInfo.GetStreamByID(vapID, temperature, pressure, mvf,
    enthalpy, component)
check = kValues.DefineVaporStreamComponentRates(component)

check = kValues.GetKValues(result)
nComp = kValues.GetNoOfComponents

End If

End Sub

```

ENGINEERING UNIT CONVERSIONS

The interfaces described in this section are used for engineering unit conversions. Beyond the interfaces in this section, only quantities in CHEMCAD 5 internal units can be used as arguments. Sometime the same quantities are needed in current user units. Therefore three interfaces are defined for engineering unit conversions.

Description IStreamUnitConversion interface offers the engineering unit conversion for stream data between CHEMCAD 5 internal units and current user units. It also gives the engineering unit strings for table formatting in Excel.

Definition

```

dispinterface IStreamUnitConversion
{
methods:
    short DefineStreamInCurUserUnit(float temp, float pres, float enth,
        float flow, short flowOption, VARIANT compFlow);

    short GetStreamInInternalUnit(float* tempR, float* presPsia, float*
        enthBtu_Hr, float* rateLbmol_Hr, VARIANT compFlowLbmol_Hr);

    short GetStreamInCurUserUnit(float* temp, float* pres, float* enth,
        float* tMoleRate, float* tMassRate, float* tStdLVolRate, float*
        tStdVVolRate, VARIANT compFlow);

    short DefineStreamInInternalUnit(float tempR, float presPsia, float
        enthBtu_Hr, VARIANT compFlowLbmol_Hr);

    void GetCurUserUnitString(BSTR* tempUnit, BSTR* presUnit, BSTR*
        enthUnit, BSTR* tMoleRateUnit, BSTR* tMassRateUnit, BSTR*
        tStdLVolRateUnit, BSTR* tStdVVolRateUnit, BSTR* compUnit, BSTR*
        compCate);
};

```

Methods

It takes two methods calls to complete a unit conversion for stream data. The first method defines a set of stream data and the following method call returns the converted stream data to the caller.

To convert stream data in current user units, user first call

Method DefineStreamInCurUserUnit
Returns number of data items received from array compFlow

Argument	Type	Description
temp	float	temperature [in]
pres	float	pressure [in]
enth	float	enthalpy [in]
flow	float	total flowrate [in]
flowOption	short	definition of total flowrate [in]
compFlow	float[]	component flowrates or compositions [in]

Note that the argument flowOption identifies the meaning of argument flow. It may be one the four choices 1 – mole rate; 2 – mass rate; 3 – standard liquid volume rate; 4 – standard vapor volume rate. After defining a set of stream data, user calls

Method GetStreamInInternalUnit
Returns nombre of data items returned in array compFlowLbmol_Hr

Argument	Type	Description
tempR	float	temperature [out]
presPsia	float	pressure [out]
enthBtu_Hr	float	enthalpy [out]
rateLbmol_Hr	float	total flowrate [out]
compFlowLbmol_Hr	float[]	component flowrates [out]

The next pair of methods is for conversion of stream data in internal units to current user units.

Method DefineStreamInInternalUnit
Returns number of data items received from array compFlowLbmol_Hr

Argument	Type	Description
tempR	float	temperature [in]
presPsia	float	pressure [in]
enthBtu_Hr	float	enthalpy [in]
compFlowLbmol_Hr	float[]	component flowrates [in]

Method GetStreamInCurUserUnit
Returns number of data items returned in array compFlow

Argument	Type	Description
temp	float	temperature [out]
pres	float	pressure [out]
enth	float	enthalpy [out]
tMoleRate	float	total mole rate [out]
tMassRate	float	total mass rate [out]
tStdLVolRate	float	total standard liquid volume rate [out]
tStdVVolRate	float	total standard vapor volume rate [out]
compFlow	float[]	component flowrates or compositions [out]

When tabulating the stream data, one more method can be called to get the engineering unit strings of the current user units.

Method GetCurUserUnitString

Argument	Type	Description
tempUnit	BSTR	temperature unit string[out]
presUnit	BSTR	pressure unit string [out]
enthUnit	BSTR	enthalpy unit string [out]
tMoleRateUnit	BSTR	total mole rate unit string [out]
tMassRateUnit	BSTR	total mass rate unit string [out]
tStdLVolRateUnit	BSTR	total standard liquid volume rate unit string [out]
tStdVVolRateUnit	BSTR	total standard vapor volume rate unit string [out]
compUnit	BSTR	component data unit string [out]
compCate	BSTR	component data category [out]

The argument compCate gives the representation of component data in the current user units, i.e., either Flowrates or Components. In the case of Flowrates, the units are given by compUnit. If it is Components, the definition of the compositions is given by compUnit. For example, the compCate returns Components, compUnit returns mole fractions indicating the component data in current user units are in mole fractions.

Description IUnitOpSpecUnitConversion interface exposes methods for engineering unit conversions of the unit specification parameters. The unit specifications are organized as a one-dimensional array of float type. User should prepare this array along with the second array for receiving converted data before calling the methods of this interface.

Definition

```
dispinterface IUnitOpSpecUnitConversion
{
methods:
    short FromInternalUnitsToCurUserUnits(VARIANT specInInternal,
        VARIANT specInCurUser);

    short FromCurUserUnitsToInternalUnits(VARIANT specInCurUser,
        VARIANT specInInternal);

    short GetUnitOpSpecArrayDimension();
    short GetCurUserUnitString(short unitOpID, short parIndex, BSTR*
        unitString, BSTR* paramName);

    short GetNoOfParameters(short unitOpID);
};
```

Methods One method need to be called when converting data from current user units to internal units

Method FromCurUserUnitsToInternalUnits
Returns number of data items returned

Argument	Type	Description
specInCurUser	Float[]	Unit specifications in current user units [in]
specInInternal	Float[]	Unit specifications in internal units [out]

On the other hand, when converting data from internal units to current user units, user calls

Method FromInternalUnitsToCurUserUnits
Returns number of data items returned

Argument	Type	Description
specInInternal	Float[]	Unit specifications in current user units [in]
specInCurUser	Float[]	Unit specifications in internal units [out]

Following the call to FromInternalUnitsToCurUserUnits, user can get the parameter name and unit string by calling

Method GetCurUserUnitString
Returns number of data item returned

Argument	Type	Description
unitOpID	short	Unit ID [in]
parIndex	short	Unit specification array index (base one) [in]
unitString	BSTR	Parameter engineering unit string
paramName	BSTR	Name of the parameter

or call

Method GetNoOfParameters
Returns number of parameters used by the given unit

to find out the actual number of parameters occupied by the specifications for the unit.

Description IEngUnitConversion interface is a general engineering conversion utility. It performs unit conversions between CHEMCAD 5 internal units and the current user units. The unit categories are defined on the following table.

EngUnit ID	Description	Abbreviation
1	Molar flow rate	MOLE
2	Temperature	TEMP
3	Temperature difference	DELTAT
4	Pressure	PRES
5	Pressure difference	DELTAP
6	Enthalpy rate	QRATE
7	Work rate	WORK
8	Area	AREA
9	Heat transfer coefficient	HTC
10	Heat of reaction	HREAC
11	Length	LEN
12	Diameter	DIA
13	Liquid density	LDEN
14	Viscosity	VISC
15	Surface tension	ST
16	Mass flow rate	MASS
17	Crude flow rate	CRUDE
18	Cake resistance	CAKE
19	Solubility	SOL
20	Specific volume	SVOL
21	Dipole moment	DMOMENT
22	Vapor density	VDEN
23	Volume	VOL
24	Velocity	VEL
25	Medium resistance	MRES
26	Packed column pressure drop	PACKDP
27	Specific heat capacity (mass base)	WHEAT
28	Thermal conductivity	TCOND
29	Liquid volume rate	LVRATE
30	Vapor volume rate	VVRATE
31	Mole (no time)	WTMOLE
32	Mass (no time)	WTMASS
33	Heat (no time)	HEAT
34	Enthalpy/Mole	MOLEHC
35	Enthalpy/Mole or Enthalpy/Mass	SPHC
36	Enthalpy/Mass	MHEAT
37	Fouling factor (reserved)	FOULF
38	Heat capacity mole or mass bases	HEATCAP
39	Inverse liquid volume	ILV
40	Time	TIME

Definition

```

dispinterface IEngUnitConversion
{
methods:
    short FromInternalUnitToCurUserUnit(short engUnitID, float
        valueInInternal, float* valueInUser, BSTR* userUnitString);

    short FromCurUserUnitToInternalUnit(short engUnitID, float
        valueInUser, float* valueInInternal);
};

```

Methods

Two methods are exposed for unit conversion of an engineering quantity between CHEMCAD 5 internal units and the current user units. To convert a value in CHEMCAD 5 internal units to user units, use the following method.

Method FromInternalUnitToCurUserUnit
Returns number of value returned

Argument	Type	Description
engUnitID	short	Engineering unit category ID [in]
valueInInternal	float	Value in CC5 internal unit [in]
valueInUser	float	Value in current user unit [out]
userUnitString	BSTR	User unit string [out]

To convert a value in current user units to CHEMCAD 5 internal units, user calls

Method FromCurUserUnitToInternalUnit
Returns number of value returned

Argument	Type	Description
engUnitID	short	Engineering unit category ID [in]
valueInUser	float	Value in CC5 internal unit [in]
valueInInternal	float	Value in current user unit [out]

Example

This subroutine generates a table of stream data for outlets in current user units.

```
Sub PrintATableOfOutletStreams(ByVal ChemCADEntry As Object)
On Error Resume Next

' generate a table of outlet stream data in current user units

' get chemCAD objects
Dim curUnitOp As Object
Dim strInfo As Object
Dim strConv As Object

Set curUnitOp = ChemCADEntry.GetCurUnitOp
Set strInfo = ChemCADEntry.GetStreamInfo
Set strConv = ChemCADEntry.GetStreamUnitConversion

'inlets
Dim check As Integer
Dim nOutlets As Integer
nOutlets = curUnitOp.GetNoOfOutlets
Dim outletIDs(1 To SIZE_STREAM_ARRAY) As Integer
check = curUnitOp.GetOutletIDs(outletIDs)

' setup table header
Dim tempUnit As String
Dim presUnit As String
Dim enthUnit As String
Dim moleRateUnit As String
Dim massRateUnit As String
Dim stdLRateUnit As String
Dim stdVRateUnit As String
Dim compUnit As String
Dim compCategory As String

check = -1
strConv.GetCurUserUnitString tempUnit, presUnit, enthUnit, moleRateUnit,
massRateUnit, stdLRateUnit, stdVRateUnit, compUnit, compCategory
```

```

Dim outletSheet As Worksheet
Set outletSheet = Worksheets("OUTSTREAMS")
outletSheet.Activate

outletSheet.Cells(STREAM_ROW_SECTION_TITLE, STREAM_COL_NAMES).value =
"Outlet Streams"
outletSheet.Cells(STREAM_ROW_ID, STREAM_COL_NAMES).value = "Stream IDs"
outletSheet.Cells(STREAM_ROW_LABEL, STREAM_COL_NAMES).value = "Labels"

outletSheet.Cells(STREAM_ROW_TEMPERATURE, STREAM_COL_NAMES).value =
"Temperature"
outletSheet.Cells(STREAM_ROW_TEMPERATURE, STREAM_COL_UNITS).value =
tempUnit

outletSheet.Cells(STREAM_ROW_PRESSURE, STREAM_COL_NAMES).value =
"Pressure"
outletSheet.Cells(STREAM_ROW_PRESSURE, STREAM_COL_UNITS).value = presUnit

outletSheet.Cells(STREAM_ROW_ENTHALPY, STREAM_COL_NAMES).value =
"Enthalpy"
outletSheet.Cells(STREAM_ROW_ENTHALPY, STREAM_COL_UNITS).value = enthUnit

outletSheet.Cells(STREAM_ROW_MOLE_FRACTION, STREAM_COL_NAMES).value =
"Vapor Mole Fraction"

outletSheet.Cells(STREAM_ROW_MOLE_FLOWRATE, STREAM_COL_NAMES).value =
"Total Mole FlowRate"
outletSheet.Cells(STREAM_ROW_MOLE_FLOWRATE, STREAM_COL_UNITS).value =
moleRateUnit

outletSheet.Cells(STREAM_ROW_MASS_FLOWRATE, STREAM_COL_NAMES).value =
"Total Mass FlowRate"
outletSheet.Cells(STREAM_ROW_MASS_FLOWRATE, STREAM_COL_UNITS).value =
massRateUnit

outletSheet.Cells(STREAM_ROW_STD_LIQ_VOLRATE, STREAM_COL_NAMES).value =
"Total Std. Liq. Vol. FlowRate"
outletSheet.Cells(STREAM_ROW_STD_LIQ_VOLRATE, STREAM_COL_UNITS).value =
stdLRateUnit

outletSheet.Cells(STREAM_ROW_STD_VAP_VOLRATE, STREAM_COL_NAMES).value =
"Total Std. Vap. Vol. FlowRate"
outletSheet.Cells(STREAM_ROW_STD_VAP_VOLRATE, STREAM_COL_UNITS).value =
stdVRateUnit

outletSheet.Cells(STREAM_ROW_COMPFLOW_VALUE, STREAM_COL_NAMES).value =
compCategory
outletSheet.Cells(STREAM_ROW_COMPFLOW_VALUE, STREAM_COL_UNITS).value =
compUnit

'intletSheet.Cells(STREAM_ROW_STD_VAP_VOLRATE, STREAM_COL_VALUE + 1).Value
= TotalStdVVolFlowRate()

Dim compIndex As Integer
Dim compCount As Integer
Dim curRow As Integer
Dim compName As String
Dim compID As Integer
curRow = STREAM_ROW_COMPFLOW_VALUE
compCount = strInfo.GetNoOfComponents

For compIndex = 1 To compCount Step 1
    compName = strInfo.GetComponentNameByPosBaseOne(compIndex)
    compID = strInfo.GetComponentIDByPosBaseOne(compIndex)

    outletSheet.Cells(curRow + compIndex, STREAM_COL_NAMES).value =
    compName
    outletSheet.Cells(curRow + compIndex, STREAM_COL_UNITS).value = compID
Next compIndex

' prepare flash data

```

```

Dim tempR As Single
Dim presPsia As Single
Dim vapFrac As Single
Dim enthBtu_Hr As Single
Dim compLbmol_Hr(1 To SIZE_COMP_ARRAY) As Single

Dim pressure As Single
Dim enthalpy As Single
Dim temperature As Single
Dim mvf As Single
Dim moleRate As Single
Dim massRate As Single
Dim stdLRate As Single
Dim stdVRate As Single
Dim compRate(1 To SIZE_COMP_ARRAY) As Single

Dim streamIndex As Integer
Dim id As Integer
Dim idcopy As Integer
Dim col As Integer
Dim curCol As Integer
curCol = STREAM_COL_VALUE

Dim valKeeper As Integer

For streamIndex = 1 To nOutlets Step 1
    valKeeper = streamIndex
    id = outletIDs(streamIndex)
    idcopy = id

    check = strInfo.GetStreamByID(id, tempR, presPsia, vapFrac,
    enthBtu_Hr, compLbmol_Hr)
    check = strConv.DefineStreamInInternalUnit(tempR, presPsia,
    enthBtu_Hr, compLbmol_Hr)
    check = strConv.GetStreamInCurUserUnit(temperature, pressure,
    enthalpy, moleRate, massRate, stdLRate, stdVRate, compRate)

    col = curCol + valKeeper
    outletSheet.Cells(STREAM_ROW_ID, col).value = outletIDs(valKeeper)
    outletSheet.Cells(STREAM_ROW_LABEL, col).value =
    strInfo.GetStreamLabelByID(idcopy)

    outletSheet.Cells(STREAM_ROW_TEMPERATURE, col).value = temperature
    outletSheet.Cells(STREAM_ROW_PRESSURE, col).value = pressure
    outletSheet.Cells(STREAM_ROW_ENTHALPY, col).value = enthalpy
    outletSheet.Cells(STREAM_ROW_MOLE_FRACTION, col).value = vapFrac

    outletSheet.Cells(STREAM_ROW_MOLE_FLOWRATE, col).value = moleRate
    outletSheet.Cells(STREAM_ROW_MASS_FLOWRATE, col).value = massRate
    outletSheet.Cells(STREAM_ROW_STD_LIQ_VOLRATE, col).value = stdLRate
    outletSheet.Cells(STREAM_ROW_STD_VAP_VOLRATE, col).value = stdVRate

    For compIndex = 1 To compCount Step 1
        outletSheet.Cells(curRow + compIndex, col).value =
    compRate(compIndex)
    Next compIndex

    streamIndex = valKeeper
Next streamIndex

End Sub

```

PHYSICAL PROPERTIES OF PURE COMPONENTS

Description This interface provides the connection to retrieve pure component data for any component on current component list. The following component physical properties are made available.

Property ID	Physical Property of Pure Component	Units
1	Molecular weight	
2	Critical temperature	R
3	Critical pressure	Psia
4	Acentric factor	
5	Normal boiling point	R
6	Specific gravity	
7	Ideal gas heat of formation	Btu/lbmol
8	Ideal gas Gibbs free energy of formation	Btu/lbmol

Definition

```
dispinterface ICompPPData
{
  methods:
    short GetDataInInternalUnit(short compPos, short compPPID, float*
    value);
};
```

Methods

Method GetDataInInternalUnit
Returns number of item returned
(0 – pos or/and id is invalid; 1 – value found)

Argument	Type	Description
CompPos	short	Component position [in]
CompPPID	short	Stream property ID [in]

Example

REFERENCE

[1] CHEMCAD 5 user-added module manual.