

CHEMCAD 5 SERVICES TO VISUAL BASIC APPLICATIONS

User Manual

Chemstations, Inc.

INTRODUCTION

CHEMCAD 5 offers three approaches to link with Visual Basic applications or other Windows applications. In the first approach CHEMCAD 5 is activated as an out-of-process server from a Visual Basic application. This enables a VBA application, such as Excel macros, to delegate computational tasks to CHEMCAD 5 running in background. The second approach functions through the addition of a new unit operation in CHEMCAD 5 called "Excel Unit". With this unit operation, a user can program specific tasks in VBA macros within an Excel workbook, and CHEMCAD 5 can execute these macros in the context of a unit operation during a simulation run. The third approach facilitates user-specified rate expressions in use with CHEMCAD 5 built-in kinetic reactor (KREA) and batch reactor (BREA).

CHEMCAD 5.x publishes a set of COM interfaces to two type libraries, cc5.tlb and ccx2xls.tlb. Through these interfaces, many CHEMCAD 5.x functions and data are made available to other Windows applications. By executing the interface methods, a user can retrieve flowsheet topology and its unit operations, update stream data and unit operation parameters, calculate k-values and enthalpy, flash streams and convert engineering units. This document presents a detailed discussion on the uses of these published interfaces within user-developed programs. We assume that user is familiar with VBA syntax and has some knowledge of Microsoft COM. The knowledge on process modeling with CHEMCAD 5.x is necessary.

VB SERVER

In this approach, CHEMCAD 5 plays the role of a simulation engine, and the user application organizes the calculation activities. The user application presents a user interface for any user instructions on the calculations, while CHEMCAD 5 carries out the instructions on the background. The user application is often developed in MS Excel using VBA programs. It is responsible for activating CHEMCAD 5 as a COM object. After activating CHEMCAD 5, the user application should proceed with its calculation tasks that may in turn be delegated to CHEMCAD 5.

To activate CHEMCAD 5, a user application executes VBA command `CreateObject("CHEMCAD.VBServer")`, where "CHEMCAD.VBServer" is the programmatic identifier (ProgID) for CHEMCAD 5. This command returns a handle to an object interface called `ICHEMCADVBServer`. Through the methods on this interface, the user application can further access many other CHEMCAD 5 object interfaces. In this section, we discuss interface `ICHEMCADVBServer` and its methods. This interface is published in `CC5.tlb`.

Description `ICHEMCADVBServer` interface provides simulation job related operations and serves as an entry point to all other interfaces. Through this interface, a user application can find a simulation job, load one of its cases, update stream data or unit operation parameters, and run steady state or dynamic simulations.

A handle to this interface is often created using its programmatic identifier "CHEMCAD.VBServer" in a VBA macro. In the following example, a handle to `ICHEMCADVBServer` of CHEMCAD 5 is passed to VB variable `CC5`.

```

Dim CC5 As Object

Sub LoadCC5()
    Set CC5 = CreateObject("CHEMCAD.VBServer")
End Sub

```

...

Definition:

```

dispinterface ICHEMCADVBServer
{
methods:

    boolean LoadJob(BSTR* bstrJobPath);
    boolean RunJob();

    // interfaces defined in ccx2xls.tlb
    VARIANT GetFlowsheet();
    VARIANT GetUnitOpInfo();
    VARIANT GetUnitOpSpecUnitConversion();
    VARIANT GetStreamInfo();
    VARIANT GetStreamUnitConversion();
    VARIANT GetStreamProperty();
    VARIANT GetKValues();
    VARIANT GetEnthalpy();
    VARIANT GetFlash();
    VARIANT GetEngUnitConversion();
    VARIANT GetCompPPData();

    // simulation job operations
    BSTR GetWorkDir();
    boolean SwitchWorkDir(BSTR* bstrNewWorkDir);
    short GetNoOfJobsInWorkDir();
    BSTR GetJobAt(short jobIndex);
    short GetNoOfCasesInJob(BSTR* jobName);
    BSTR GetCaseAt(BSTR* jobName, short caseIndex);

    // simulations
    short SSRunAllUnits();
    short SSRunSelectedUnits(VARIANT unitIDs);
    short DynamicRunAllSteps();
    short DynamicRunStep();
    void DynamicRestoreToInitialState();
    float GetDynamicTimeInMinute();
    float GetDynamicTimeStepInMinute();
    short GetSimulationMode();
    void TakeProcessSnapShot(BSTR* pathname);
    void LoadProcessSnapShot(BSTR* pathname);

    void PauseSimulation();
    void SetCurrentAsInitialState();
    float OTSGetTimeScale();
    short OTSSetTimeScale(float timeScale);
    BSTR ShowRunTimeMessages();
    short SSRunOptimization(short runMode);
    short GetAppVersion();
    short EditUnitOpPar(short unitOpID);
};

```

Methods

Each of the following methods exposed by ICHEMCADVBServer takes a void argument and returns a handle to another interface to the calling subroutine. All the interfaces are published in ccx2xls.tlb and discussed in detail later in this document.

<i>Method</i>	GetFlowsheet
<i>Returns</i>	a handle to IFlowsheet

<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>	GetStreamInfo
<i>Returns</i>	a handle to IStreamInfo
<i>Method</i>	GetStreamUnitConversion
<i>Returns</i>	a handle to IStreamUnitConversion
<i>Method</i>	GetStreamProperty
<i>Returns</i>	a handle to IStreamProperty
<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>	GetFlash
<i>Returns</i>	a handle to IFlash
<i>Method</i>	GetEngUnitConversion
<i>Returns</i>	a handle to IEngUnitConversion
<i>Method</i>	GetCompPPData
<i>Returns</i>	a handle to ICompPPData

The following methods defined on this interface can be used to switch current work directory, locate a simulation job, and load one of its cases.

<i>Method</i>	LoadJob
<i>Returns</i>	a boolean flag, TRUE if a job is loaded

Argument	Type	Description
bstrJobPath	BSTR*	Job path, e.g., "c:\cc5data\job1\case1.ccx"

<i>Method</i>	GetWorkDir
<i>Returns</i>	a string of work directory
<i>Method</i>	SwitchWorkDir
<i>Returns</i>	a boolean flag, TRUE if a job is loaded

Argument	Type	Description
bstrNewWorkDir	BSTR*	a string of work directory

<i>Method</i>	GetNoOfJobsInWorkDir
<i>Returns</i>	count of jobs in work directory
<i>Method</i>	GetJobAt
<i>Returns</i>	a string of job name

Argument	Type	Description
jobIndex	int	job index in job array

Method GetNoOfCasesInJob
Returns count of cases in job directory

Argument	Type	Description
jobName	String	job name for which cases are seeked

Method GetCaseAt
Returns a string of case name

Argument	Type	Description
jobName	String	job name for which cases are seeked
caseIndex	Int	case index

After a job is loaded, the following functions can be called for steady state or dynamic simulations.

Call SSRunAllUnits to simulation entire flowsheet, and use SSRunSelectedUnits to run selected set of units on the flowsheet. Use SSRunOptimization to perform optimizations or parameter estimations with data reconciliation.

Method SSRunAllUnits
Returns 0 - normal return, otherwise if error occurs.
1 - No license for VB server, functional argument type mismatch, or recycle loop did not converge
<0 - Is the number of flowsheet errors times -1

Method SSRunSelectedUnits
Returns 0 - normal return, otherwise if error occurs.
1 - No license for VB server, functional argument type mismatch, or recycle loop did not converge
<0 - Is the number of flowsheet errors times -1

Method SSRunOptimization
Returns 0 - normal return, otherwise if error occurs.

Argument	Type	Description
unitIDs	Integer	Array of unitop ids.
runMode	Integer	0 – Optimization; 1 – Data reconciliation & parameter estimation

The following routines are dynamic simulation functions. DynamicRunAllSteps will simulate the flowsheet from the current time to the end of integration, while DynamicRunStep simply advances one time step from the current time. Use DynamicRestoreToInitialState to reset the flowsheet to its initial state. The current time and time step can be retrieved by calling GetDynamicTimeInMinute and GetDynamicTimeStepInMinute respectively.

Method DynamicRunAllSteps
Returns 0 - normal return; otherwise if error occurs.

Method DynamicRunStep
Returns 0 - normal return; otherwise if error occurs.

Method DynamicRestoreToInitialState

Method GetDynamicTimeInMinute
Returns Current dynamic time in minutes

Method GetDynamicTimeStepInMinute
Returns Current dynamic time step in minutes.

During dynamic simulations, user can update the initial state to the current state by calling SetCurrentAsInitialState.

Method SetCurrentAsInitialState

For operator training applications, the time axis may be scaled to observe normal or slow motion of the plant state changes. The next two functions are designed to access the time scale factor.

Method OTGetTimeScale
Returns Current time scale factor.

Method OTSetTimeScale

Argument	Type	Description
timeScale	Single	New value of time scale factor (1 to 10)

Before running any simulation functions listed above, the simulation mode set in CHEMCAD 5 for the currently loaded simulation job can be determined using

Method GetSimulationMode()
Returns Current mode, 0 - steady state, 1 - dynamic.

The following function can be used to stop a simulation run in both dynamic or steady state mode.

Method PauseSimulation()

User can retrieve runtime messages from CHEMCAD using ShowRunTimeMessages.

Method ShowRunTimeMessages()
Returns Current runtime messages

A call to the following method will instruct CHEMCAD 5 to perform a simulation based on its setting, i.e., run all units in steady state, and run one step or all steps in dynamic simulation as previously specified.

Method RunJob
Returns a boolean flag, TRUE if recycle is converged.

Method TakeProcessSnapShot
Method LoadProcessSnapShot

Both of the above functions take a file path string as a parameter. A path string should have the following format: Drive:/directory/subdirectory/.../filename. Any extension attached will be ignored. Those functions should be used in dynamic simulations in between calls to DynamicRunStep function.

<i>Method</i>	GetAppVersion
<i>Returns</i>	CHEMCAD version number (for example, 5400).

<i>Method</i>	EditUnitOpPar
---------------	---------------

This function opens the dialog for the specified unit operations on your flowsheet.

EXCEL UNIT

A versatile and user-friendly unit operation, "Excel Unit", is available in CHEMCAD 5. As all other CHEMCAD 5 unit operations, this unit has a dedicated unit icon, and it may have user-defined parameters and unit operation dialog. This unit serves as a link between CHEMCAD 5 and an Excel workbook. When placing an Excel Unit icon on a flowsheet, specifying the path to an Excel workbook and a list of macros, and running the flowsheet, CHEMCAD loads the specified Excel workbook into an instance of Excel and runs the given list of VBA macros in the Excel 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 created in CHEMCAD 5 is passed to the macro 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.

Before the Excel Unit is made available, many user-specific tasks can only be accomplished using Calculators or User-Added Modules in CHEMCAD 5. Compared to these unit operations, the Excel unit has several advantages. Since Excel unit delegates all calculations to its Excel workbook, it does not require any compilation and linking with CHEMCAD 5. A user develops VBA macros in an Excel workbook. An Excel unit can have up to 10 macros. A user can break down the calculation task into smaller pieces, and implement them in several macros. The testing is also straightforward. VBA is an interpretive language, breakpoints can be placed at any part of the macros, and the values of the variables can be monitored during the execution.

The steps of specifying 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, i.e., 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 paths of unit dialog files, the Excel workbook and the names of the macros to be executed during a simulation run. Third, program VBA macros in the specified Excel workbook and set breakpoints for debugging. After finishing all previous steps, run simulations from CHEMCAD 5, and 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 upon the first opening of Excel Unit dialog, 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 for 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 are required to have identical name. These files are necessary for entering parameters of the Excel Unit. These parameters can later be retrieved within the Excel macros during simulation runs. User can find more information on creating unit specification dialog files (.my, .lab, .map) in the Screen Builder documentation [1].

The default Excel workbook created by CHEMCAD 5 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.

For the new users, simple editing of the VBA macros can be done within the Excel Unit dialog. When a user opens the Excel Unit dialog and selects VB Sub tab, a portion of the macro ExclUop will appear. In this macro, we have retrieved the inlet and outlet data, and equipment parameters to VB variables. A user can use these variables directly in the subsequent calculations. The common procedure is to calculate the outlets from given inlets and equipment parameters. Following this portion, the outlets will be flashed with the given flash mode. The user macros in the given Excel workbook will only be updated when the user marks the option “Write to Excel workbook”, otherwise CHEMCAD 5 will only update the source code in .bas file. This facility should be used with caution, and users should be aware that CHEMCAD 5 will not compile the VB code and the code shown in the dialog will not reflect the changes made directly in the Excel workbook using Visual Basic Editor.

The interface ICHEMCADEntry for Excel Unit is one that will lead to all other interfaces. The only role of this interface is to allow user to reach other interfaces within their macros. Only after getting hold of other interfaces can a user use any ChemCAD 5 calculations. Therefore, each of the VBA macros should have exactly one argument, i.e., Sub UserSub (ByVal ChemCADEntry As Object), if it will be activated from CHEMCAD 5.

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 ICHEMCAEntry
{
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);
    BSTR GetCurJobPath();
    BSTR GetCurJobName();
    BSTR GetProgPath();

    BSTR GetWorkPath();
    BSTR GetPoolPath();
};

```

Methods

Each of the methods exposed by ICHEMCAEntry 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

Method GetCompPPData
Returns 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.

Method GetSaveWorkBookFlag
Returns a boolean flag

Method SetSaveWorkBookFlag
Returns a boolean flag

Argument	Type	Description
bSaveAfterRun	boolean	CHEMCAD saves workbook if set TRUE

The following two functions allow a user to get the current job directory and case name. It may be useful if you are handling files from your VBA codes. Function GetCurJobPath returns a full path of job directory, e.g., "C:\cc5data\my project".

Method GetCurJobPath
Returns a string containing current job directory.

Method GetCurJobName
Returns a string containing currently loaded job name under job directory

Method GetProgPath
Returns a string containing CHEMCAD program directory

Method GetWorkPath
Returns a string containing work directory

Method GetPoolPath
Returns a string containing pool directory

USER RATE EXPRESSIONS

Chemical reaction rate expressions may take various forms, and in many cases they cannot fit to the Arrhenius rate equations in CHEMCAD 5. CHEMCAD 5 allows the user to use built-in reactors with the user-specified rate expressions documented in an Excel workbook. If user is modeling a reaction system that deviate from the standard rate expressions in CHEMCAD 5, the user may open a kinetic reactor or a batch reactor equipment dialog, select "user-specified" for kinetic rate. After entering stoichiometric coefficients, user rate expression dialog will appear. Here a user specifies a VB file to hold the generated codes of user rate expressions, and an Excel workbook for CHEMCAD 5 to activate at the time of simulation. Then user will enter their rate expressions for each of the chemical reactions. When click "OK", the VB codes will be generated and stored in the specified VB file. Next CHEMCAD 5 imports the codes in the VB file into the Excel workbook directly. User is recommended to enter Visual Basic Editor, check for any errors using Debug, and make necessary changes.

When composing a rate expression, CHEMCAD 5 offers liquid concentrations and partial vapor pressures of components, temperature and pressure, etc. If needed, user can access CHEMCAD 5 information through ICHEMCADVBServer interface passed to every rate expression macro as an argument.

During a simulation run, CHEMCAD 5 will activate the user rate expressions from either a kinetic reactor or a batch reactor. If the same set of rate expressions is used in several reactors, enter all the expressions in the equipment dialog for one reactor, and specify the identical Excel workbook path for the other reactors. If different sets of rate expressions are used in different reactors, specify different Excel workbook paths, VB files and rate expressions for each reactor.

The Excel macro activated during simulation has the following signature:

```
Sub UserRxnRates(ByRef Rate() As Double, ByVal Temp As Double, ByVal Pres As Double, ByVal Rpm As Double, ByVal RVol As Double, ByVal LFrc As Double, ByRef Conc() As Double, ByRef PPri() As Double, ByRef KPar() As Single, ByVal ChemCADEntry As Object)

    Rate(1) = RxnRate001(Temp, Pres, Rpm, Conc, KPar, ChemCADEntry)
    Rate(2) = RxnRate002(Temp, Pres, Rpm, Conc, KPar, ChemCADEntry)

    ...

End Sub
```

The array **Rate()** returns the rates of chemical reactions to a CHEMCAD 5 reactor, **Temp** is current temperature in user unit, **Pres** is current pressure in user unit, **Rpm** is the propeller speed, **RVol** is reactor volume, **LFrc** is the current liquid volume fraction in the reactor, array **Conc()** contains concentrations in user units, array **Ppri()** contains the partial pressures, array **Kpar()** carries kinetic parameters, and **ChemCADEntry** is an object handle through which many CHEMCAD 5 information can be accessed. Currently 10 kinetic parameters are assigned for each reaction. Among the 10 parameters, the first is frequency factor and the second is the exponential term including activation energy and temperature. The remaining parameters of the 10 are reserved for CHEMCAD at this time. Within this macro, the functions for each of the rate expressions are called. The rate function for reaction no. 1 looks like the following

```
Function RxnRate001(ByVal Temp As Double, ByVal Pres As Double, ByVal Rpm As Double, ByVal RVol As Double, ByVal LFrc As Double, ByRef Conc() As Double, ByRef PPri() As Double, ByRef KPar() As Single, ByVal ChemCADEntry As Object) As Double

    ...

    RxnRate001 = ...

End Function
```

CHEMCAD 5 initially generates these codes. A user can make changes as needed. Additional information can be accessed through ChemCADEntry object handle. The interface for this object handle is described in the previous section on Excel Unit.

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 GetReactorLiquidPhase(VARIANT compLbmol);
    short GetReactorVaporPhase(VARIANT compLbmol);
    short GetRerunFlowsheetFlag();
    short SetRerunFlowsheetFlag(short rerunIfOne);
    short SetRunTimeMessage(VARIANT msg);
};
```

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

If the current unit is kinetic reactor (KREA) or batch reactor (BREA) and user specifies rate expressions, user may retrieve liquid (include solid) and vapor phase from the following functions.

Method GetReactorLiquidPhase / GetReactorVaporPhase
Returns count of data items returns

Argument	Type	Description
compLbmol	float[]	Float array containing liquid or vapor amount

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

Users can pass text messages to CHEMCAD during runtime to be shown in trace window by passing a VB string or an array of VB strings to the following function in user-supplied macros.

Method SetRunTimeMessage
Returns number of message strings retrieved

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);

```

```

        short GetDynamicTime(float* dynTime, BSTR* timeUnit);
};

```

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]

Method GetDynamicTime
Returns number of items returned

Argument	Type	Description
dynTime	float*	current dynamic time [out]
timeUnit	BSTR*	engineering unit of dynamic time [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 in a process. The data may be used in operational decisions or computational procedures. 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);
    short ReflashStream(short streamID);
    short PutStreamInCurUserUnitByID(short streamID, short flashMode,
        float temp, float pres, float enth, float mvf, float tRate,
        short tRateOption, VARIANT compRate);
    short GetStreamInCurUserUnitByID(short streamID, float* temp,
        float* pres, float* enth, float* mvf, float* tMoleRate, float*
        tMassRate, float* tStdLVolRate, float* tStdVVolRate, VARIANT
        compRate);

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

    short OTSGetStreamPar(float* parVal, short parID, short streamID);
    short OTSPutStreamPar(short streamID, short parID, float parVal);
    short OTSGetStreamParInUserUnit(float* parVal, short parID, short
        streamID);
    short OTSPutStreamParInUserUnit(short streamID, short parID, float
        parVal);

    boolean GetStreamCost(short streamID, short* costType, float*
        costVal);

    boolean SetStreamCost(short costType, float costVal, short
        streamID);

Function GetParticleSizeDistribution(ByVal streamID As Integer, ByVal solidCompID
    As Integer, ByVal noOfCuts As Integer, ByRef particleSizes() As Single, ByRef
    weightFractions() As Single) As Integer

Function PutParticleSizeDistribution(ByVal streamID As Integer, ByVal solidCompID
    As Integer, ByVal noOfCuts As Integer, ByVal particleSizes() As Single, ByVal
    weightFractions() As Single) As Integer

Function GetSolids(ByRef solidCompIDs() as Integer) As Integer
```

Function PutSolidFlag(ByVal nSize As Integer, ByVal solidCompIDs() As Integer, ByVal bSetFlag As Integer) As Integer

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]

Method ReflashStream (short streamID);
Returns flash result using current mode (0 – converged)

The next pair of methods allows user to retrieve stream data in current user units.

Method GetStreamInCurUserUnitByID
Returns number of data items returned in array compRate

Argument	Type	Description
streamID	integer	stream id
temp	float	temperature [out]
pres	float	pressure [out]
enth	float	enthalpy [out]
mvf	float	vapor mole fraction [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]
compRate	float[]	component flowrates or compositions [out]

Method PutStreamInCurUserUnitByID
Returns number of data items received from array compRate

Argument	Type	Description
streamID	integer	stream ID
flashMode	integer	flash calculation mode [in]
temp	float	temperature [in]
pres	float	pressure [in]
enth	float	enthalpy [in]
mvf	float	mole vapor fraction [in]
tRate	float	total flowrate [in]
tRateOption	short	definition of total flowrate [in]
compFlow	float[]	component flowrates or compositions [in]

Note that the argument flashMode defines the flash applied to the stream data, 1 – TP flash; 2 – VP flash; 3 – VT flash; 4 – HP flash; otherwise no flash calculation. The argument tRateOption 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.

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.

The following functions can be used in a steady state simulation as well as during a dynamic simulation. For a complete list of stream parameters, please refer to the next section (IStreamProperty interface).

Method OTSGetStreamPar/ OTSGetStreamParInUserUnit
 OTSPutStreamPar/ OTSPutStreamParInUserUnit
Returns number of data items received from or passed to CHEMCAD

Argument	Type	Description
parVal	Float	Current parameter value [in/out]
parID	Integer	parameter ID [int]
streamID	Integer	stream ID [in]

Method GetStreamCost / SetStreamCost
Returns True if successful.

Argument	Type	Description
costVal	Float	Actual cost [out/in]
costType	Integer	Type of cost [out/in]
streamID	Integer	stream ID [in]

Method GetParticleSizeDistribution
Returns 1 if successful, 0 if data cannot be found.

Argument	Type	Description
streamID	Integer	stream ID [in]
solidCompID	Integer	Solid component ID [in]
noOfCuts	Integer	Count of data points [out]
particleSize	Float[]	Particle size array [out]
weightFractions	Float[]	Weight fraction at each size [out]

Method PutParticleSizeDistribution
Returns
 0 Specified component cannot be found or has no distribution
 1 Success
 2 Invalid function arguments
 3 Invalid dimensions of arrays
 4 Invalid size of array

5 Arrays are not SafeArray of Single
 6 Internal error

Argument	Type	Description
streamID	Integer	stream ID [in]
solidCompID	Integer	Solid component ID [in]
noOfCuts	Integer	Count of data points [out]
particleSize	Single()	Particle size array [=] microns [out]
weightFractions	Single()	Weight fraction at each size [out]

Method GetSolids
Returns

Number of solids in streams if successful

Less than zero return on error

-1 Function argument error

-2 Memory error

-3 Internal error

-4 Destination array is too small

Argument	Type	Description
solidCompIDs	Integer()	CHEMCAD ids for components [out]

Method PutSolidFlag
Returns Number of components set or reset by this function

Argument	Type	Description
nSize	Integer	Number of components changing status [in]
solidCompIDs	Integer()	CHEMCAD ids of components [in]
bSetSolid	Integer	Flag indicating to set status to solid 1 or set status to nonsolid 0 for specified components [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	ft3/hr

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

Property ID	Stream Property Description (vapor)	Units
26	Vapor mole rate	lbmol/h

27	Vapor mass rate	lb/h
28	Vapor enthalpy	Btu/h
29	Vapor entropy	Btu/F/h
30	Vapor Mw	--
31	Vapor actual density	lb/ft3
32	Vapor actual volume rate	ft3/h
33	Vap std. Liquid volume rate	ft3/h
34	Vap std. Vapor volume rate	scfh
35	Vapor cp	Btu/lbmol-F
36	Vapor Z factor	--
37	Vapor viscosity	cP
38	Vapor thermal conductivity	Btu/h-ft-F
39	Cp/Cv	--

Property ID	Stream Property Description (liquid)	Units
41	Liquid mole rate	lbmol/h
42	Liquid mass rate	lb/h
43	Liquid Enthalpy	Btu/h
44	Liquid Entropy	Btu/F/h
45	Liquid Mw	--
46	Liquid actual density	lb/ft3
47	Liquid actual volume rate	ft3/h
48	Liquid std. Liquid volume rate	ft3/h
49	Liquid std. Vapor volume rate	scfh
50	Liquid cp	Btu/lbmol-F
51	Liquid Z factor	--
52	Liquid viscosity	cP
53	Liquid thermal conductivity	Btu/h-ft-F
54	Liquid surface tension	Dyne/cm
55	Liquid and solid actual density	lb/ft3
56	Liquid and solid actual volume	ft3/h
57		

Property ID	Stream Property Description (solid)	Units
60	Solid mole rate	lbmol/h
61	Solid mass rate	lb/h
62	Solid Mw	--
63	Solid Enthalpy	Btu/h
64	Solid cp	Btu/lbmol-F
65	Solid actual volume	ft3/h
66	Solid density	lb/ft3
67	Solid std. Vapor volume rate	scfh
68	Solid std. Liquid volume rate	ft3/h

Definition

```
dispinterface IStreamProperty
{
methods:
```

```

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

float GetStreamProperty(short propID);
short GetStreamPropertiesInUserUnits(short streamID, VARIANT
streamProp, VARIANT propUserUnits);
};

```

Methods

To calculate stream properties, a 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]

Method GetStreamPropertiesInUserUnits
Returns number of data items returned fro array streamProp

Argument	Type	Description
streamID	short	Stream ID [in]
streamProp	float[60]	Stream property array [out]
propUserUnits	BSTR[60]	User units of the property [out]

Example

UNIT OPERATION SPECIFICATION DATA

Each unit operation on a flowsheet is specified by assigning values to a set of parameters. This set of parameters is 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);

    // column specific functions
    short GetNoOfStagesByID(short unitOpID);
    short GetColumnHydraulicsFromSizing(short unitOpID, BSTR
        columnType, VARIANT holdupLbmol, VARIANT pressurePsia,
        VARIANT floodPercent);
    short SetColumnPressureProfile(short unitOpID, VARIANT
        pressurePsia);
    short GetColumnStageData(short unitOpID, short stageID, VARIANT
        vaporLbmol_Hr, VARIANT liquidLbmol_Hr, float* temperatureR,
        float* pressurePsia);

    // individual equipment parameter
    short GetUnitOpPar(float* parVal, short parID, short unitOpID);
    short PutUnitOpPar(short unitOpID, short parID, float parVal);
    short GetUnitOpParInCurUserUnit(float* parVal, short parID, short
        unitOpID);
    short PutUnitOpParInCurUserUnit(short unitOpID, short parID, float
        parVal);

    // batch reactor, CSTR, PFR
    short GetKineticReactionPar(float* parVal, short unitOpID, short
        rxnNo, short parID, short compIdx);
    short PutKineticReactionPar(short unitOpID, short rxnNo, short
        parID, short compIdx, float parVal);

    // EREA
    short GetEquilibriumReactionPar(float* parVal, short unitOpID,
        short rxnNo, short parID, short compIdx);
    short PutEquilibriumReactionPar(short unitOpID, short rxnNo, short
        parID, short compIdx, float parVal);
    short GetUnitOpRunTimeErrorCode(short unitOpID);
    BSTR GetUnitOpRunTimeErrorStr(short unitOpID);

    short OTSGetUnitOpPar(float* parVal, short parID, short unitOpID);
    short OTSPutUnitOpPar(short unitOpID, short parID, float parVal);
    short OTSGetUnitOpParInCurUserUnit(float* parVal, short parID,
        short unitOpID);
    short OTSPutUnitOpParInCurUserUnit(short unitOpID, short parID,
        float parVal);
};
```

Methods Each unit can store up to 250 parameters in an array of float type. The number of actual parameters can be found from the following method.

<i>Method</i>	GetUnitOpSpecArrayDimension
<i>Returns</i>	number of actual parameters for this unit operation

With a known unit ID, a 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 Op has an identity property which identifies the role of the unit Op (mixer, flash, kinetic reactor, etc). In CHEMCAD 5, this identity 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

Several functions are provided for the computational needs with CHEMCAD 5 column units, distillation towers (TOWR and SCDS), tower plus (TPLS) and liquid-liquid extractor (EXTR). These functions have no effect when called upon other types of unit operations.

Method GetNoOfStagesByID
Returns stage number for column units, 0 otherwise.

Function GetColumnHydraulicsFromSizing retrieves the data generated during equipment sizing in CHEMCAD 5. All the arrays have base 1 and size of the number of stages in the column (retrieved using GetNoOfStagesByID). The column type can be either tray or packed column, and the default is tray column.

Method GetColumnHydraulicsFromSizing
Returns -1 wrong input; 0 normal; 1 no sizing data file.

Argument	Type	Description
unitOpID	short	Unit ID [in]
columnType	BSTR	4 character string, "pack" or "tray"; either cases [in]
holdupLbmol	float[]	Array containing mole holdup on each tray [out]
pressurePsia	float[]	Array containing pressure of each tray [out]
floodPercent	float[]	Array containing flood percentage of each tray [out]

Function SetColumnPressureProfile is defined to update the column pressure profile according to the given data in array pressurePsia, This has to be an array of float, base 1 and size of the number of stages. The pressure data the array have to match the stage numbers.

Method SetColumnPressureProfile
Returns -1 wrong input; 0 normal; 1 no update.

Argument	Type	Description
unitOpID	short	Unit ID [in]
pressurePsia	float[]	Array containing pressure of each tray [in]

Use GetColumnStageData to retrieve the simulation results of a column, one stage at a time from unit ID and stage ID. The two component arrays should have base 1 and size of 200.

Method GetColumnStageData
Returns -1 wrong input; 0 normal; 1 error.

Argument	Type	Description
unitOpID	short	Unit ID [in]
stageID	short	Stage ID, base 1 [in]
vaporLbmol_Hr	float[]	Array containing component vapor holdup [out]
liquidLbmol_Hr	float[]	Array containing component liquid holdup [out]
temperatureR	float	temperature of the stage [out]
pressurePsia	float	pressure of the stage [out]

The following routines are offered to access unit operation parameter individually. An user may retrieve or update a specific parameter either in internal units or in current user units as noted by the function names. Functions GetUnitOpPar and GetUnitOpParInCurUserUnit are used to retrieve the current parameter value in CHEMCAD 5.x, and the other two functions are used for updating a parameter value from the user application.

Method GetUnitOpPar
Returns 0 parameter not found; 1 parameter found.

Method PutUnitOpPar
Returns 0 parameter not found; 1 parameter found.

Method GetUnitOpParInCurUserUnit
Returns 0 parameter not found; 1 parameter found.

Method PutUnitOpParInCurUserUnit

Returns 0 parameter not found; 1 parameter found.

Argument	Type	Description
unitOpID	short	Unit ID [in]
parID	short	Parameter ID, base 1 [in]
parVal	float	Parameter value if found

The following routines are designed specifically for reactors offered by CHEMCAD 5. The first two GetKineticReactionPar and PutKineticReactionPar can be used on batch reactors, CSTRs, and PFRs. The next two GetEquilibriumReactionPar and PutEquilibriumReactionPar are for EREA unit. The return values for all routines are defined in the same way, i.e., 0 – OK, 1 – data file not found, 2 – reaction ID error, 3 – component ID error, 4 – parameter ID error.

The parameters available for kinetic reactions are given below.

Par ID	Kinetic Reaction Parameter	Need comID?
1	Frequency factor	No
2	Activation energy	No
3	Exponential factor	Yes
4	Beta factor	No
5	Adsorption frequency factor	Yes
6	Adsorption energy factor	Yes
7	Adsorption exponential factor	Yes
8	Heat of reaction	No

Method GetKineticReactionPar
Returns Integer flag.

Method PutKineticReactionPar
Returns Integer flag.

Argument	Type	Description
unitOpID	short	Unit ID [in]
rxnNo	short	Reaction number, base 1 [in]
parID	short	Parameter ID, base 1 [in]
complx	short	Component position, base 1 [in]
parVal	float	Parameter value if found

The parameters available for equilibrium reactions are given below.

Par ID	Equilibrium Reaction Parameter	Need compID
1	A factor	No
2	B factor	No
3	Heat of reaction	No
4	Approach DT	No
5	Fractional approach	No
6	Fractional conversion	No
7	Exponential factor	Yes

Method GetEquilibriumReactionPar
Returns Integer flag.

Method PutEquilibriumReactionPar
Returns Integer flag.

Argument	Type	Description
unitOpID	short	Unit ID [in]
rxnNo	short	Reaction number, base 1 [in]
parID	short	Parameter ID, base 1 [in]
complx	short	Component position, base 1 [in]
parVal	float	Parameter value if found

After a simulation run, user can check any run time error that may have occurred in each unit operation using the following functions

Method GetUnitOpRunTimeErrorCode
Returns Integer error flag.

Method GetUnitOpRunTimeErrorStr
Returns String containing error message.

The following functions can be used in a steady state simulation as well as during a dynamic simulation. The parameter ids for a given type of unit operation can be found from var???.sf files under CHEMCAD program directory.

Method OTSGetUnitOpPar/ OTSGetUnitOpParInUserUnit
OTSPutUnitOpPar/ OTSPutUnitOpParInUserUnit
Returns number of data items received from or passed to CHEMCAD

Argument	Type	Description
parVal	Float	Current parameter value [in/out]
parID	Integer	parameter ID [int]
unitOpID	Integer	flowsheet equipment ID [in]

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);
    float GetTotalEnthalpy();
    float GetTemperature();
    float GetPressure();
};
```

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 if 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

Method GetTotalEnthalpy
Returns total enthalpy in liquid and vapor

Method GetTemperature
Returns flash temperature

Method GetPressure
Returns flash pressure

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 DefineStream(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 DefineStream
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]

Once the methods DefineLiquidStream and DefineVaporStream are called, the Ion Rates, K Values, Activity and fugacity coefficients are calculated. User can then call any/all of 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 IKValues 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

<i>Method</i>	GetNoOfParameters
<i>Returns</i>	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	VLEN
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.

Definition

```

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

    // recent set of functions
    short GetNoComponents();
    short GetData(short compPos, short compPPID, VARIANT compPPVals);
    short PutData(short compPos, short compPPID, VARIANT compPPVals);
    short SaveUserCompData(short compPos);

    // bip functions
    short GetBIP(short compPos1, short compPos2, VARIANT bipVals);
    short PutBIP(short compPos1, short compPos2, VARIANT bipVals);
    void SaveBIP();
};
    
```

Methods

Initially, 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 at 60F	--
7	Ideal gas heat of formation	Btu/lbmol
8	Ideal gas Gibbs free energy of formation	Btu/lbmol

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]

Later, a group of functions are added for the retrieval and updating of user component physical properties. The properties current made available through this interface are listed in the following table. If you cannot find the properties of your interest, please contact Chemstations. Note that in the following functions the values of property Ids are defined differently from the Ids for function GetDataInInternalUnit.

Prop ID	Physical Property of Pure Component	Value Count	Units
	DIPPR Constant Property:		
1	Molecular weight	1	--
2	Critical temperature	1	R
3	Critical pressure	1	psia
4	Critical volume	1	ft ³
6	Melting point	1	R
9	Normal boiling point	1	R
11	Ideal gas heat of formation	1	Btu/lbmol
12	Ideal gas Gibbs free energy of formation	1	Btu/lbmol
16	Acentric factor	1	--
18	Solubility factor	1	(cal/cm ³)**0.5
19	Dipole moment	1	debyes
24	Solid heat of formation	1	kcal/mol
25	Solid Gibbs energy of formation	1	kcal/mol
	Additional Constant Property:		
33	Standard heat of vaporation	1	kcal/mol
34	Stiel polar factor	1	
35	Polar parameter	1	
36	Eps/K	1	
37	Molecular diameter	1	Angstroms
38	Watson factor	1	
39	Mean average boiling point	1	F
40	API gravity	1	--
41	Specific gravity at 60F	1	--
42	Rackett constant	1	--
43	Modified acentric factor	1	
44	UNIQUAC area parameter	1	
45	UNIQUAC volume parameter	1	
46	Wilson molar volume	1	
47	API net heating value	1	
48	API gross heating value	1	
49	Liquid volume constant	1	cc/mol
	DIPPR Equation:		
65	Solid density	6 (eq id + 5)	kmol/m ³
66	Liquid density	6 (eq id + 5)	kmol/m ³
67	Vapor pressure equation	6 (eq id + 5)	Pa
68	Heat of vaporation	6 (eq id + 5)	J/kmol
69	Solid heat capacity	6 (eq id + 5)	J/kmol-K
70	Liquid heat capacity	6 (eq id + 5)	J/kmol-K
71	Ideal gas heat capacity	6 (eq id + 5)	J/kmol-K
73	Liquid viscosity	6 (eq id + 5)	Pascal-sec
74	Vapor viscosity	6 (eq id + 5)	Pascal-sec
75	Liquid thermal conductivity	6 (eq id + 5)	W/m-K
76	Vapor thermal conductivity	6 (eq id + 5)	W/m-K
77	Surface tension	6 (eq id + 5)	N/m
	Additional Equation:		
97	Antoine vapor pressure equation	3	MmHg
98	Ideal gas heat capacity	6	
99	Liquid viscosity	2	Cp
100	Surface tension	2	N/m
101	Henry's constants	4	
102	MSRK parameters	2	
103	UNIFAC group data	2*n + 1	

Method GetData/PutData
Returns number of property values returned/received
 (0 – pos or/and property id is invalid)

Argument	Type	Description
compPos	short	Component position [in]
compPPID	short	Stream property ID [in]
compPPVals	float[]	Property values [out]

Method SaveUserCompData
Returns number of component for which data have been saved
 (0 – pos is invalid)

The next three functions allow a user to access the binary interaction parameters used in CHEMCAD. The BIPs of all selected components (CHEMCAD or user components) can be retrieved and updated through GetBIP and SetBIP, and can also be saved to file under the job directory by calling SaveBIP.

Method GetBIP/PutBIP
Returns number of BIP values returned/received for the specified pair
 (0 – pos is invalid)

Argument	Type	Description
compPos1	short	Component 1 position [in]
compPos2	short	Component 2 position [in]
bipVals	float[]	BIP values [out]

Method GetBIP/SetBIP

Example

The following subroutine fills an array with the critical pressure values for the component list.

```
Sub CritPressures(ByVal CHEMCADENTRY As Object)
    Dim objPhysProps As Object      ' Component Property object

    Dim intCompPos As Integer      ' position in component list
    Dim asngCritPresPSIA As Single ' array of critical pressures for components
    Dim sngProperty As Single      ' Property returned from objPhysProps
    Dim intNumberComponents As Single ' Number of components in list
    Dim intpropname As Integer

    Set strinfo = ChemCADEntry.GetStreamInfo
    Set objPhysProps = ChemCADEntry.GetCompPPData

    intNumberComponents = strinfo.GetNoOfComponents

    ' Property 3 is critical pressure
    intpropname = 3

    For intCompPos = 1 To intNumberComponents
        ' Get the critical pressure from CHEMCAD
        check = objPhysProps.GetDataInInternalUnit(intCompPos, intpropname,
sngProperty)

        asngCritPresPSIA = sngProperty
    Next intCompPos

End Sub
```

REFERENCES

- [1] CHEMCAD 5 Screen Builder Manual
- [2] CHEMCAD 5 User-Added Module Manual
- [3] CHEMCAD 5 Online Help