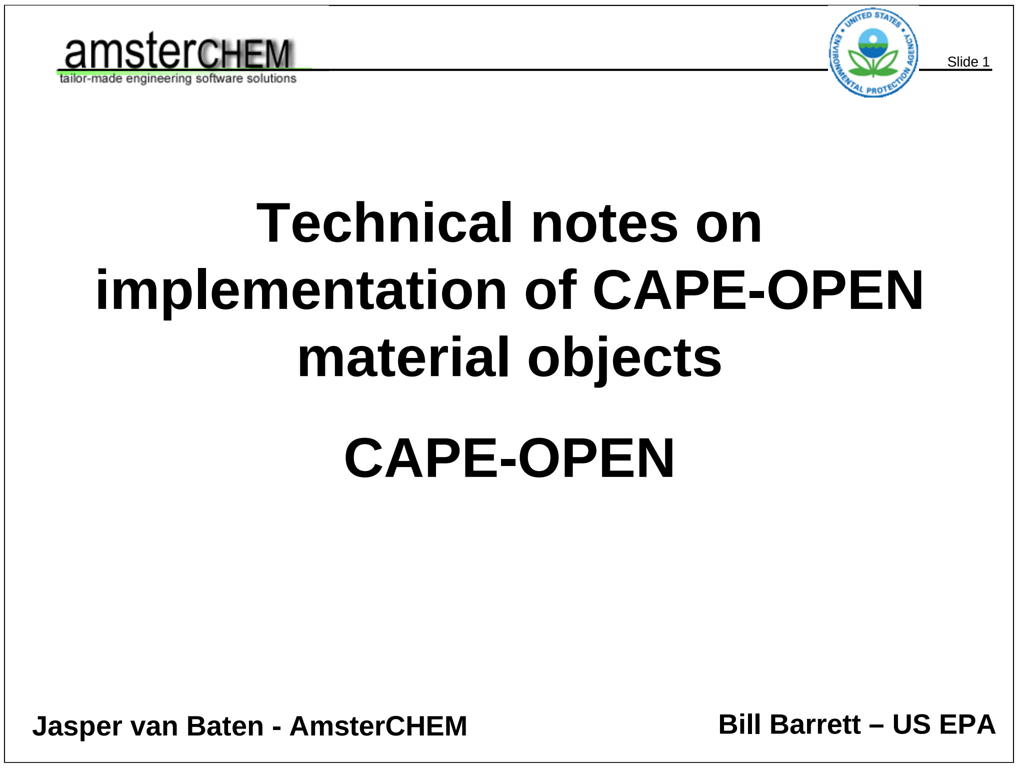

amsterCHEM
tailor-made engineering software solutions

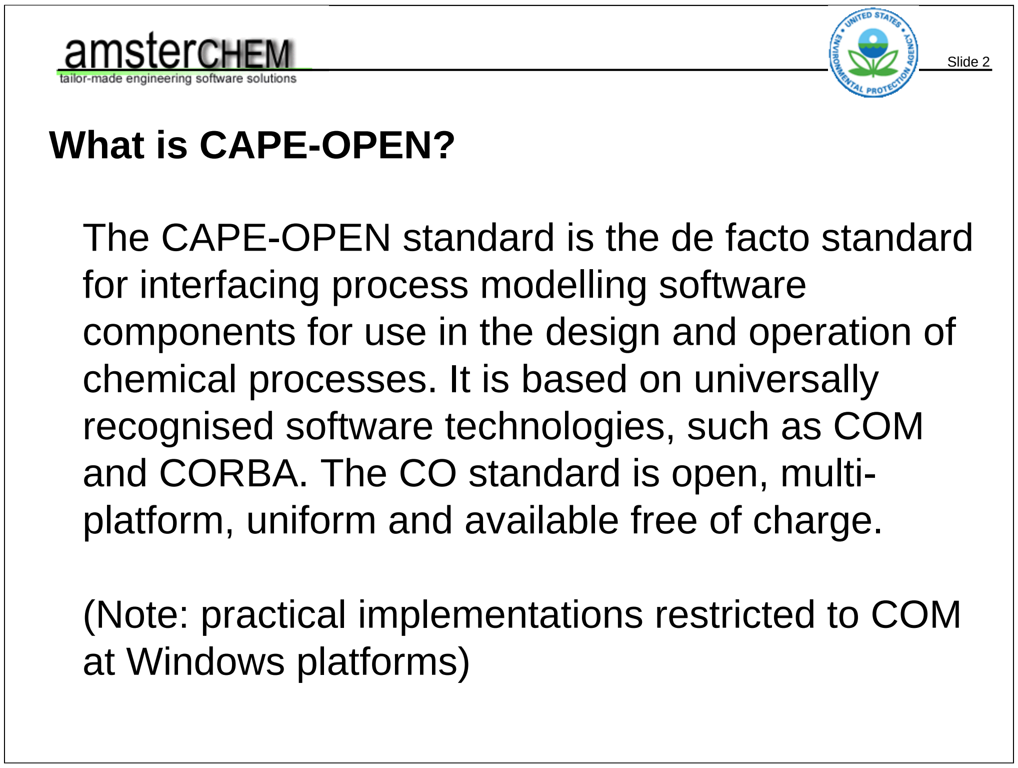
Technical notes on implementation of CAPE-OPEN material objects

CAPE-OPEN

Jasper van Baten - AmsterCHEM

Bill Barrett – US EPA

Welcome to the short course on CAPE-OPEN Material objects. These slides are from a CAPE-OPEN training session on the CAPE-OPEN European conference, May 3, Freising, Germany. Along with this training, the sample code from "MiniSim" was used, implementing a CAPE-OPEN mini PME for which the focus was on a version 1.1 CAPE-OPEN Thermo Material Object that can be used for communication with a Unit Operation as well as with a Property Package. The MiniSim example code is available to CO-LaN members.



amsterCHEM
tailor-made engineering software solutions

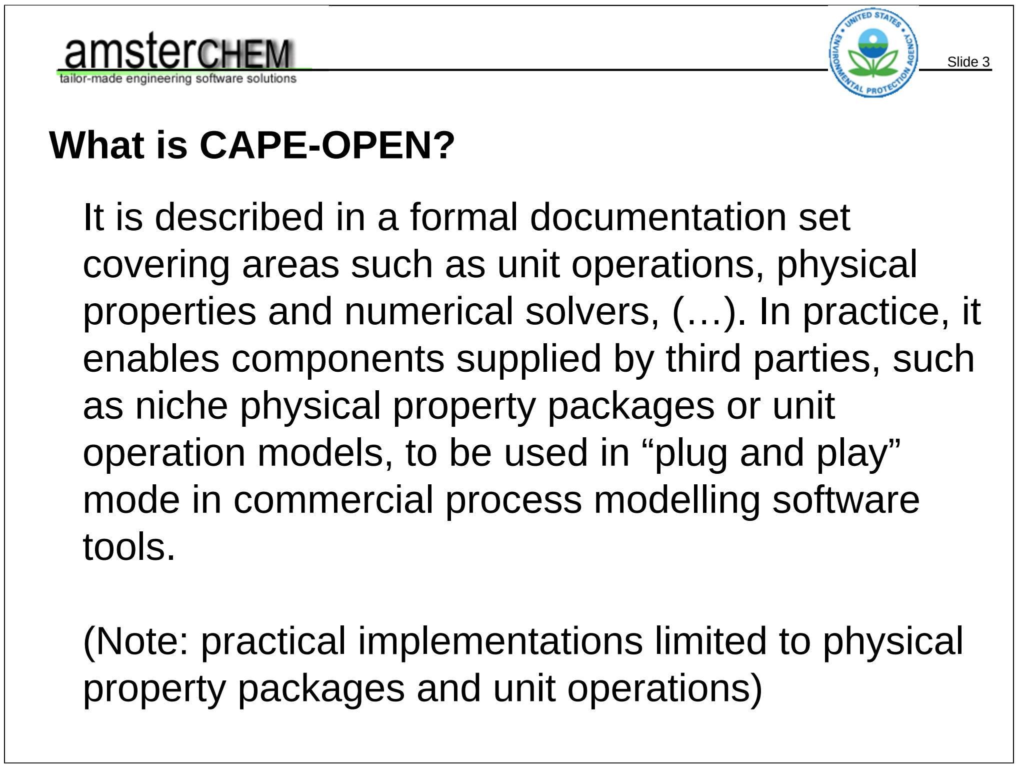
Slide 2

What is CAPE-OPEN?

The CAPE-OPEN standard is the de facto standard for interfacing process modelling software components for use in the design and operation of chemical processes. It is based on universally recognised software technologies, such as COM and CORBA. The CO standard is open, multi-platform, uniform and available free of charge.

(Note: practical implementations restricted to COM at Windows platforms)

I am sure we have all studies the CAPE-OPEN brochure. It states: “The CAPE-OPEN standard is the de facto standard for interfacing process modelling software components for use in the design and operation of chemical processes. It is based on universally recognised software technologies, such as COM and CORBA. The CO standard is open, multi-platform, uniform and available free of charge.”. It says COM and CORBA, but practically all implementations are COM based. This also diminishes its multi-platform status: COM is Windows specific, so we are looking mostly at Windows platforms.



amsterCHEM
tailor-made engineering software solutions

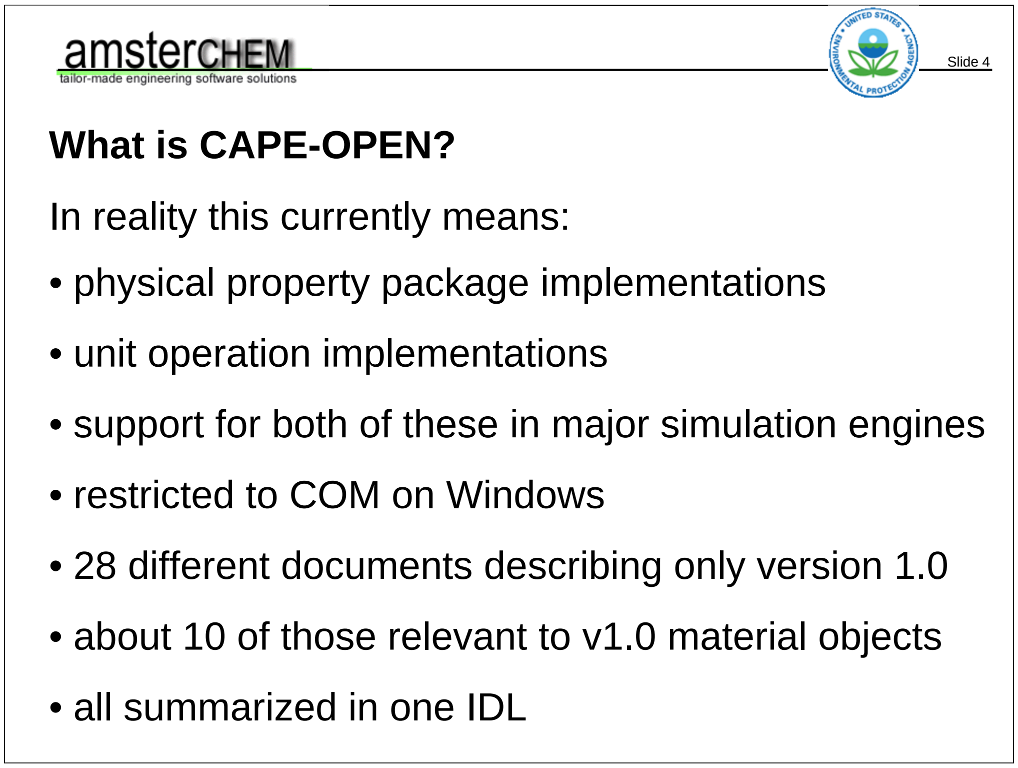
Slide 3

What is CAPE-OPEN?

It is described in a formal documentation set covering areas such as unit operations, physical properties and numerical solvers, (…). In practice, it enables components supplied by third parties, such as niche physical property packages or unit operation models, to be used in “plug and play” mode in commercial process modelling software tools.

(Note: practical implementations limited to physical property packages and unit operations)

The CAPE-OPEN brochure further states that “It is described in a formal documentation set covering areas such as unit operations, physical properties and numerical solvers, (…). In practice, it enables components supplied by third parties, such as niche physical property packages or unit operation models, to be used in “plug and play” mode in commercial process modelling software tools.”. There are one or two numerical solver implementations around, but so far I have to find people that implement a unit operation that does not have its own solvers on board. Hence, practically CAPE-OPEN in this stage means application transparent thermodynamic property ssytems and application transparent unit operations.


amsterCHEM
tailor-made engineering software solutions

Slide 4

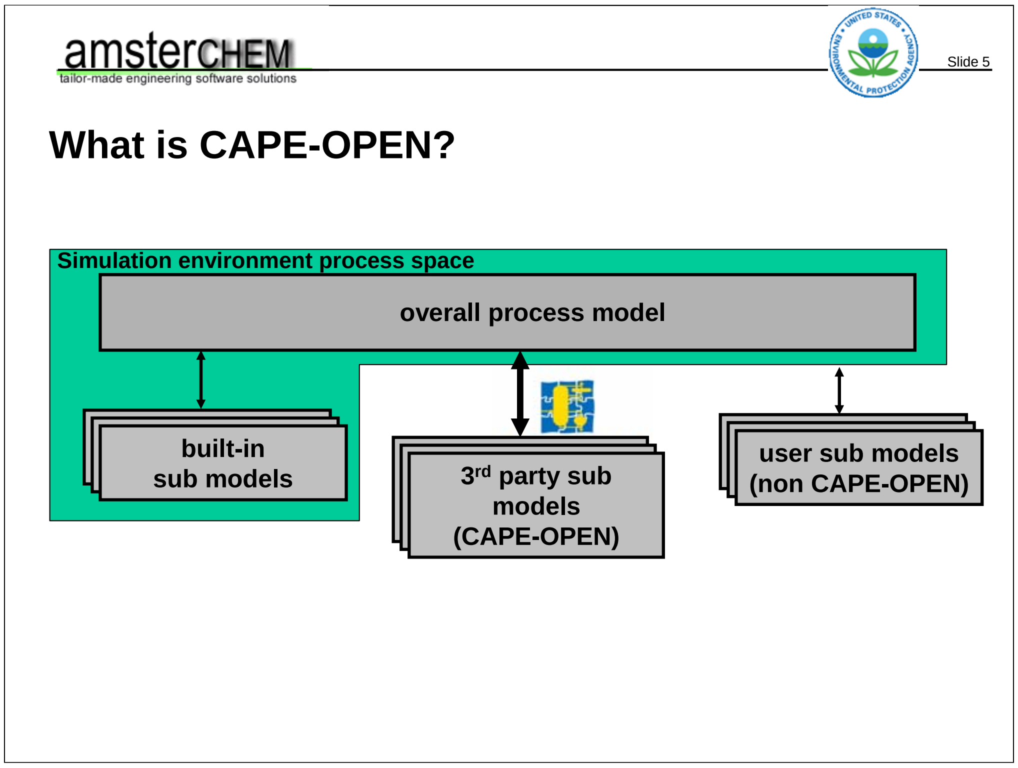
What is CAPE-OPEN?

In reality this currently means:

- physical property package implementations
- unit operation implementations
- support for both of these in major simulation engines
- restricted to COM on Windows
- 28 different documents describing only version 1.0
- about 10 of those relevant to v1.0 material objects
- all summarized in one IDL

So when we talk about CAPE-OPEN, we generally refer to physical property package implementations and unit operation implementations, as well as support for both of these in most major simulation engines. We are looking at COM based, hence Windows implementations. The documentation set (when I last counted) was 28 documents for version 1.0 alone. About 10 of these are relevant to implementation of material objects, with the thermo specification of course being the major one. Other ones include the Error common interfaces, the Unit Operation interfaces, the identification common interfaces, method & tools documents, etc.

All CAPE-OPEN definitions however are conveniently gathered in one IDL: IDL stands for Interface Definition Language, which is how software tools import the CAPE-OPEN definitions.

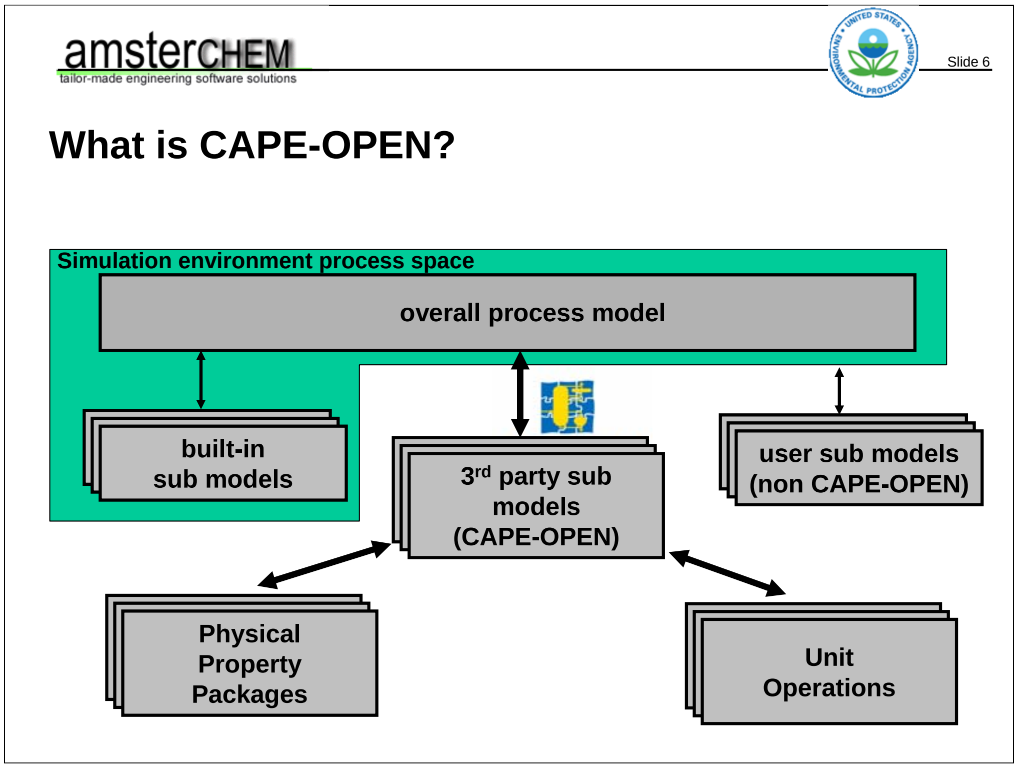

amsterCHEM
tailor-made engineering software solutions
Slide 5

What is CAPE-OPEN?

Simulation environment process space
overall process model
built-in sub models
3rd party sub models (CAPE-OPEN)
user sub models (non CAPE-OPEN)

Typically if we look at a simulation environment, we will find that there is an overall process model, such as a flowsheet. The application that does that for us, indicated in green, also typically has a large set of built-in models, such as thermo models and unit operations. And usually there is some application specific method of adding user models, shown here on the right. Often this is FORTRAN based, where the FORTRAN functions to be implemented are very specific to the application at hand.

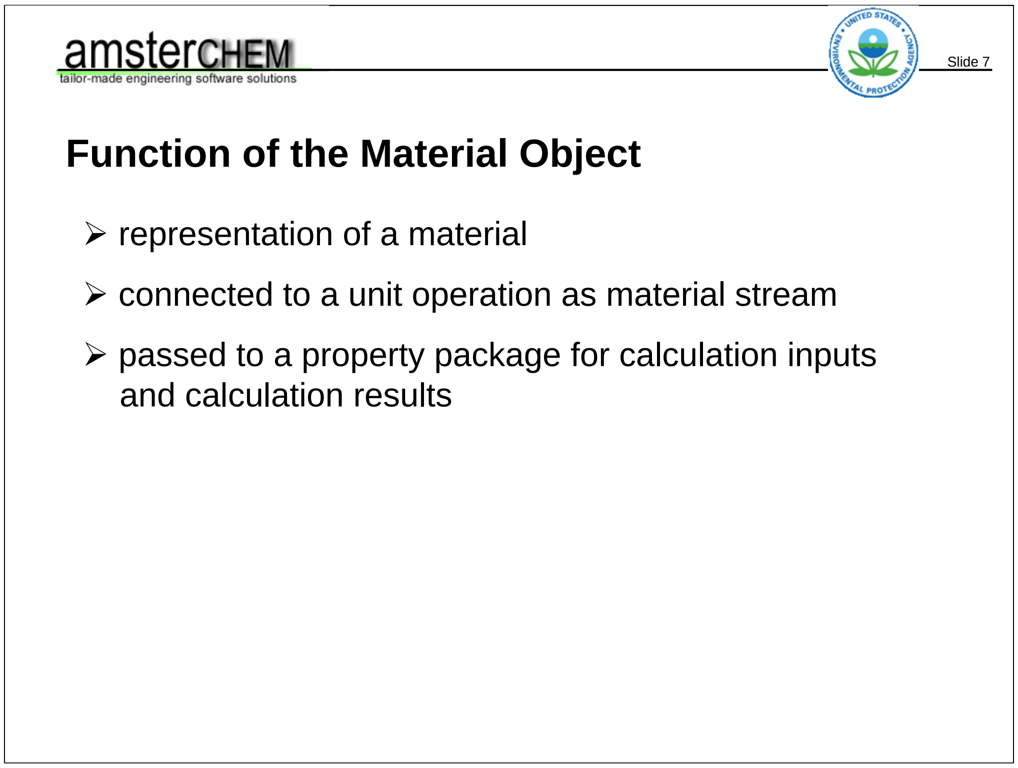
In the middle we see CAPE-OPEN as a way to add user-defined or third party models.


amsterCHEM
tailor-made engineering software solutions
Slide 6

What is CAPE-OPEN?

Simulation environment process space
overall process model
built-in sub models
3rd party sub models (CAPE-OPEN)
user sub models (non CAPE-OPEN)
Physical Property Packages
Unit Operations

Having a closer look – as we stated earlier – we will find that this applies mostly to unit operations and physical property packages. In both cases, a CAPE-OPEN material object implementation is required to talk to these. This is what we will be talking about today.


amsterCHEM
tailor-made engineering software solutions

Slide 7

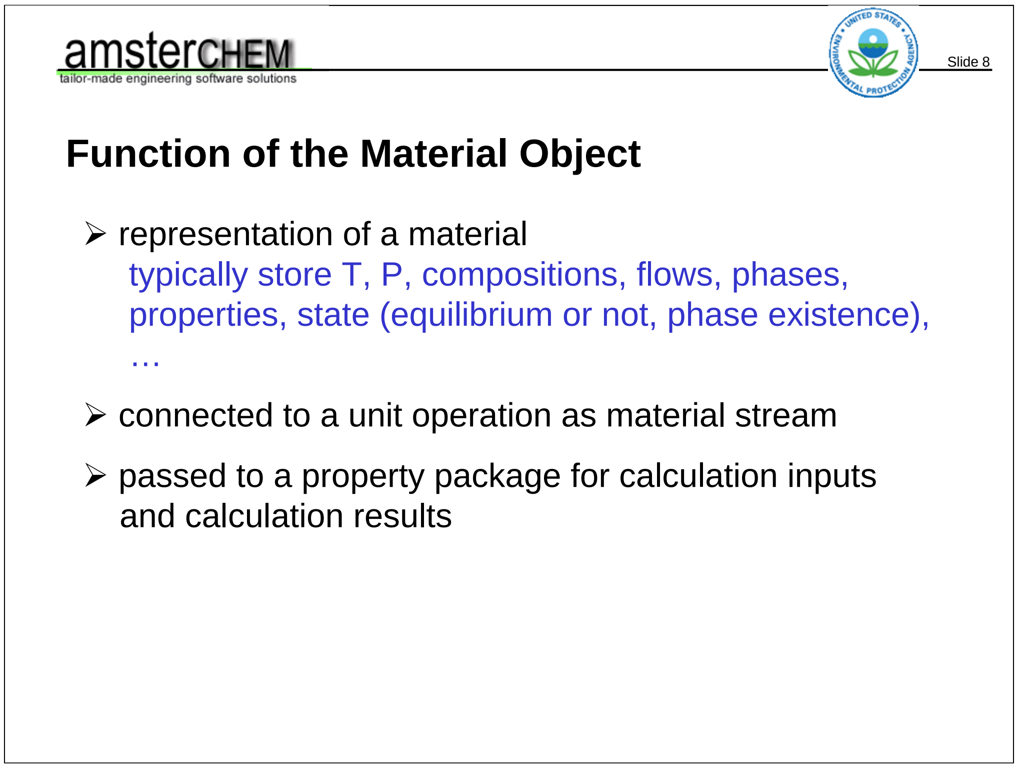
Function of the Material Object

- representation of a material
- connected to a unit operation as material stream
- passed to a property package for calculation inputs and calculation results

This short course deals with CAPE-OPEN Material Object interfaces. The Material Object interfaces lie at the heart of CAPE-OPEN communication between PMEs and PMCs. A PME as a CAPE-OPEN Process Modeling Environment. This is also referred to as COSE, CAPE-OPEN Simulation Environment. In short; simulation environment or simulation application.
A PMC is a Process Modeling Client. These represents models loaded into the simulation application. The most common PMC types are unit operations and property packages. Unit operations are representations of physical equipment (pumps, distillation columns, reactors, etc…). Property Packages are thermodynamic calculation engines that allow us to calculate physical and thermodynamic properties, to calculate thermodynamic phase equilibria, and to get details about chemical compounds and phases. We will use the term chemical compounds through-out the course. This is the term that is adopted for chemical species in CAPE-OPEN version 1.1 thermo. In CAPE-OPEN version 1.0 thermo, chemical species were called Components. This term however was dropped because of the ambiguity; the term components is also used for software components. The functionality of a Property Package is largely reflected in the functionality of a Material Object.

Both unit operation PMCs and Property Package PMCs will be used and discussed in this course.

So – the Material Object represents a Material. It is used in communication with Unit Operation PMCs as a representation of material streams connected to material ports, and it is used as property storage object in communication with a Property Package PMC.



amsterCHEM
tailor-made engineering software solutions

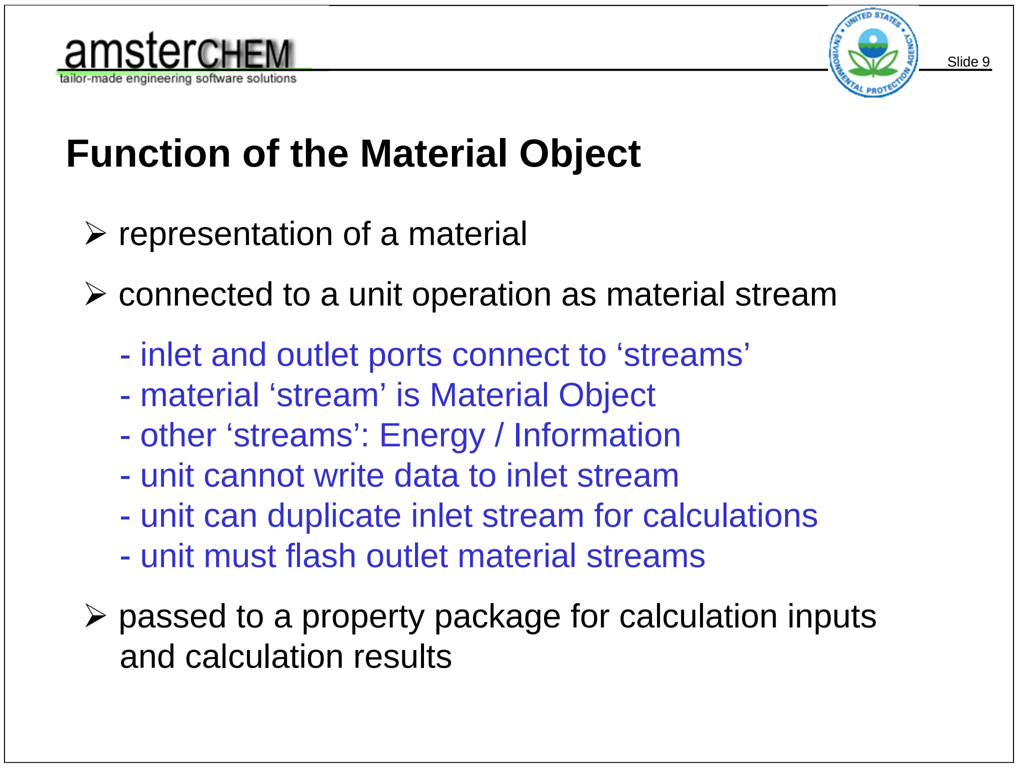
Slide 8

Function of the Material Object

- representation of a material
  typically store T, P, compositions, flows, phases, properties, state (equilibrium or not, phase existence), …
- connected to a unit operation as material stream
- passed to a property package for calculation inputs and calculation results

Thermodynamic and physical properties (with some exceptions that we will see later on) are stored on the Material Object. The special properties in CAPE-OPEN are pressure, temperature, flow, totalFlow, fraction and phaseFraction. Flow – as opposed to totalFlow – represents the flow of each compound. Fraction is the name for composition, which can be either mole fraction or mass fraction. Other than the special properties, many other properties can be stored on a material object, such as enthalpy, density, viscosity and so on. These properties have a distinct number of classifications, and depending on the CAPE-OPEN version we are using, each of these property classes has its own functions to get and set these properties.

Other than physical and thermodynamic properties, the Material object keeps track of the current state: which phases are present, and whether the material object is in thermodynamic phase equilibrium.


amsterCHEM
tailor-made engineering software solutions

Slide 9

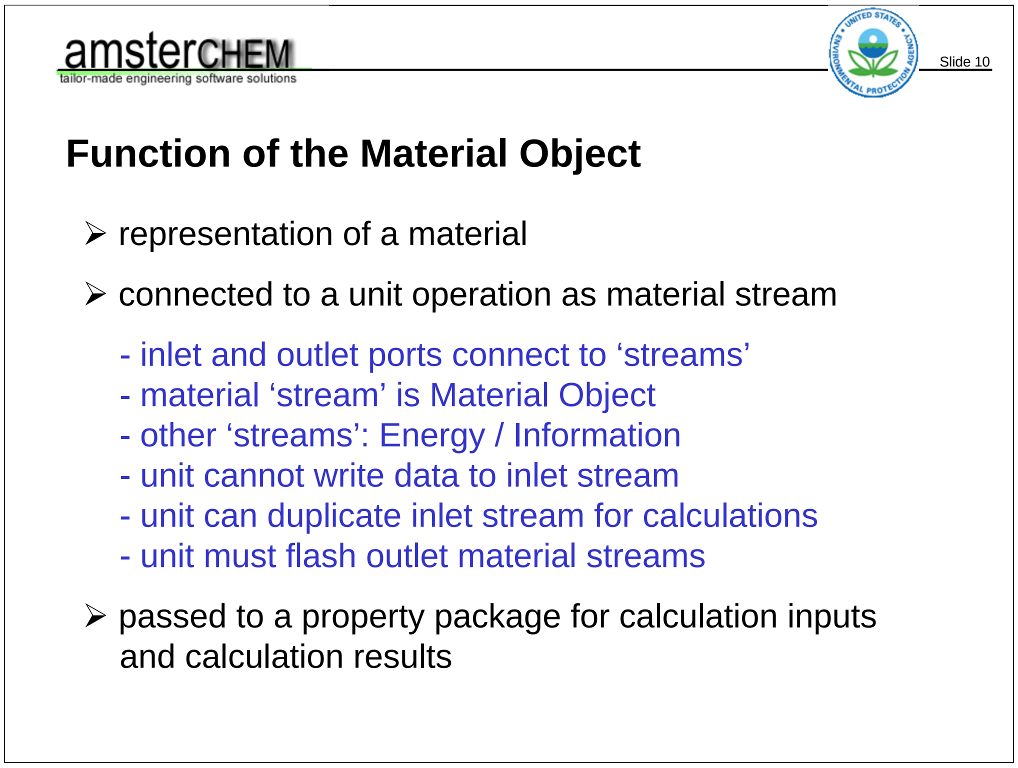
Function of the Material Object

- ➢ representation of a material
- ➢ connected to a unit operation as material stream
  - inlet and outlet ports connect to ‘streams’
  - material ‘stream’ is Material Object
  - other ‘streams’: Energy / Information
  - unit cannot write data to inlet stream
  - unit can duplicate inlet stream for calculations
  - unit must flash outlet material streams
- ➢ passed to a property package for calculation inputs and calculation results

One of the two major applications of a Material Object is to connect it to a material port of a Unit Operation. Say we have a pump unit operation, with a single inlet port and a single outlet port. The inlet port will be connected to a Material Object that contains the feed conditions of the pump. When the pump calculates, it obtains the inlet conditions. It may do some thermodynamic calculations. During its calculation, it must specify sufficient conditions for a thermodynamic equilibrium at the outlet port. The pump will set these properties on the outlet port, and then instruct the material object connected to the outlet port to perform a thermodynamic equilibrium calculation.

Unit operations may have multiple in- and outlet ports of course. It is up to the simulation application, possibly with help of user interaction, to connect the proper material objects to the proper ports. Not all ports connect to material objects however. Only material ports do. Other port types are energy and information ports. These connect to different kinds of objects, which is not part of the scope of this course. A port will – in general – check what is connected to it. A material port that expects CAPE-OPEN version 1.1 thermo may very well refuse to connect to other objects, such as version 1.0 Material Objects.

…


amsterCHEM
tailor-made engineering software solutions

Slide 10

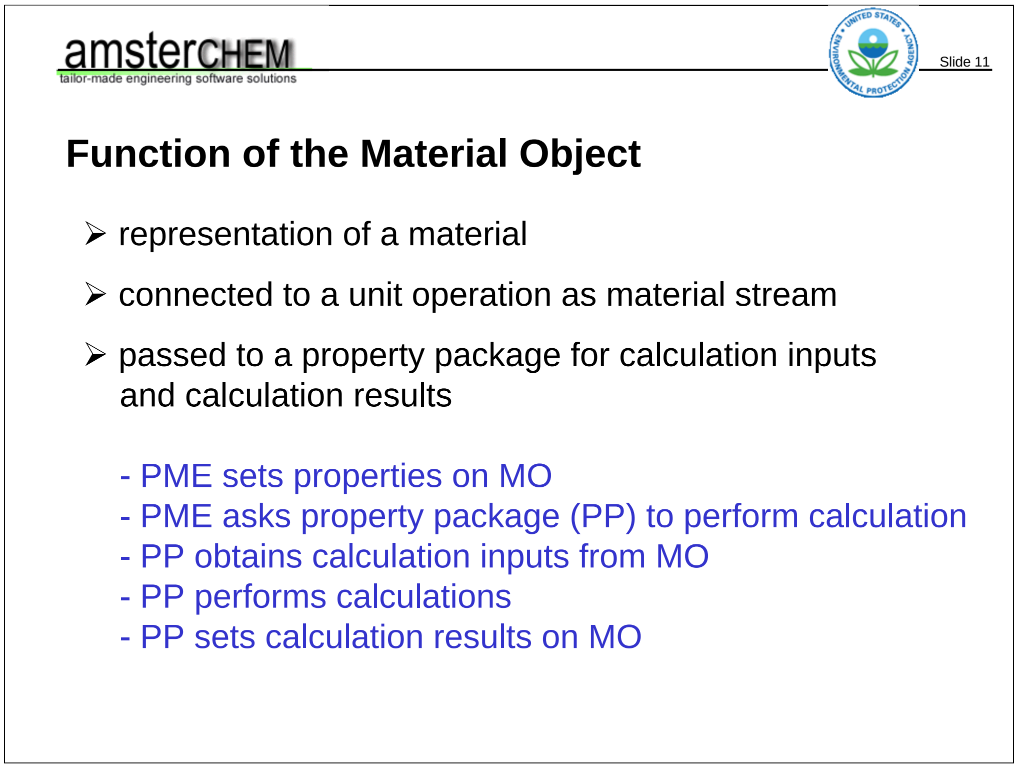
Function of the Material Object

- ➢ representation of a material
- ➢ connected to a unit operation as material stream
  - - inlet and outlet ports connect to ‘streams’
  - - material ‘stream’ is Material Object
  - - other ‘streams’: Energy / Information
  - - unit cannot write data to inlet stream
  - - unit can duplicate inlet stream for calculations
  - - unit must flash outlet material streams
- ➢ passed to a property package for calculation inputs and calculation results

…

A unit operation may not set properties or perform calculations on material objects connected to the inlet ports. It can however duplicate such a material object, and perform calculations on the duplicate. Each outlet port that is connected must be flashed by the unit operation. Not all ports may need to be connected. That depends on the unit operation. For example, a mixer unit operation could have 10 inlet ports of which one or more need to be connected. The unit operation can check valid connections during the Validate step of the unit operation.

A unit operation may be able to deal only with version 1.0 thermo, or only with version 1.1 thermo, or both. When trying to connect a material object to a material port, the unit operation attempts to get the proper interface it needs. If this interface is not present, the unit operaton may refuse the port connection.


amsterCHEM
tailor-made engineering software solutions

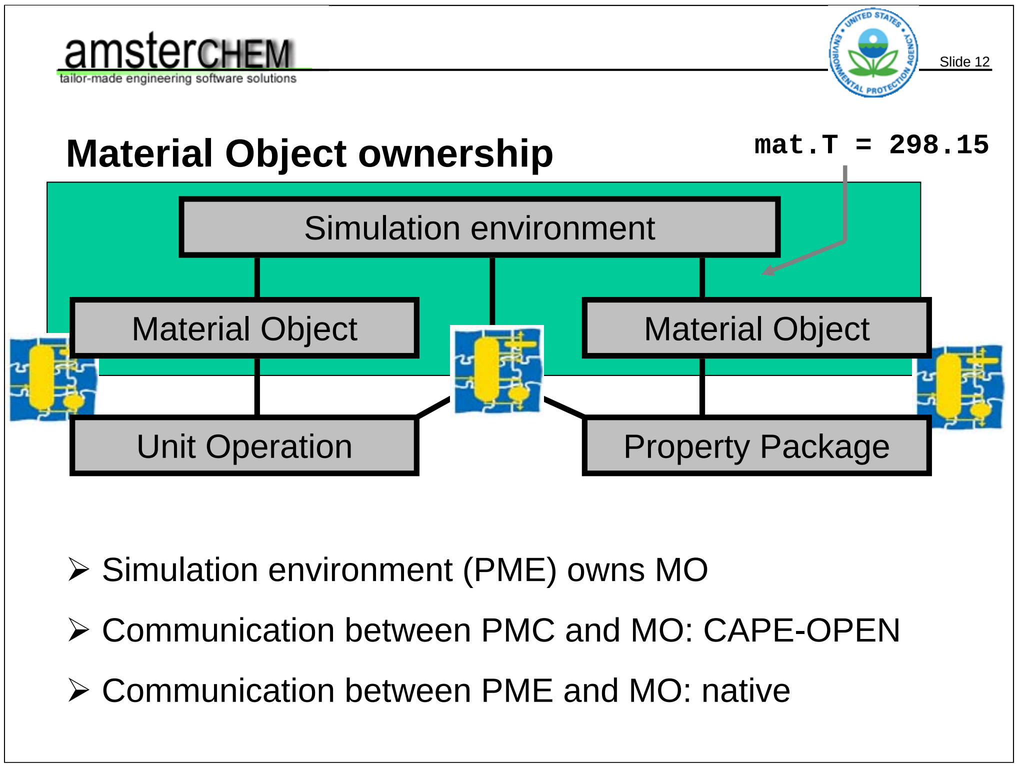
Slide 11

Function of the Material Object

- representation of a material
- connected to a unit operation as material stream
- passed to a property package for calculation inputs and calculation results

  - PME sets properties on MO
  - PME asks property package (PP) to perform calculation
  - PP obtains calculation inputs from MO
  - PP performs calculations
  - PP sets calculation results on MO

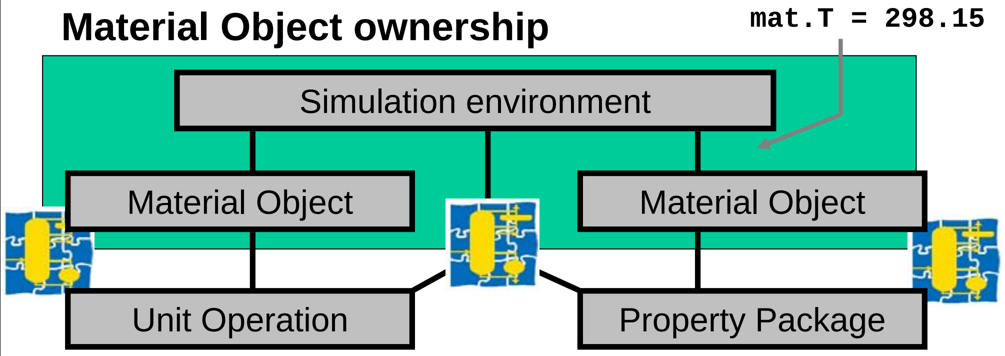
The second major use of a Material object is in communication with the Property package. The typical order of operation of a calculation using a property package is indicated here. First, the simulation application sets the required inputs for the calculation on the material object. The required inputs depends on the type of property calculation. For a vapor enthalpy calculation for example, this would be pressure, temperature and composition. Then, the simulation application asks the property package to do some calculation. With this request, it tells the Property Package which Material Object to use for that. The Property Package then obtains the required information from the property package. For the vapor enthalpy calculations, it would typically ask for which compounds are present, and which pressure, temperature and composition. Then the Property Package performs the calculation. The calculation results are stored by the Property Package on the Material Object, and control is returned to the simulation application.



amsterCHEM
tailor-made engineering software solutions

Slide 12

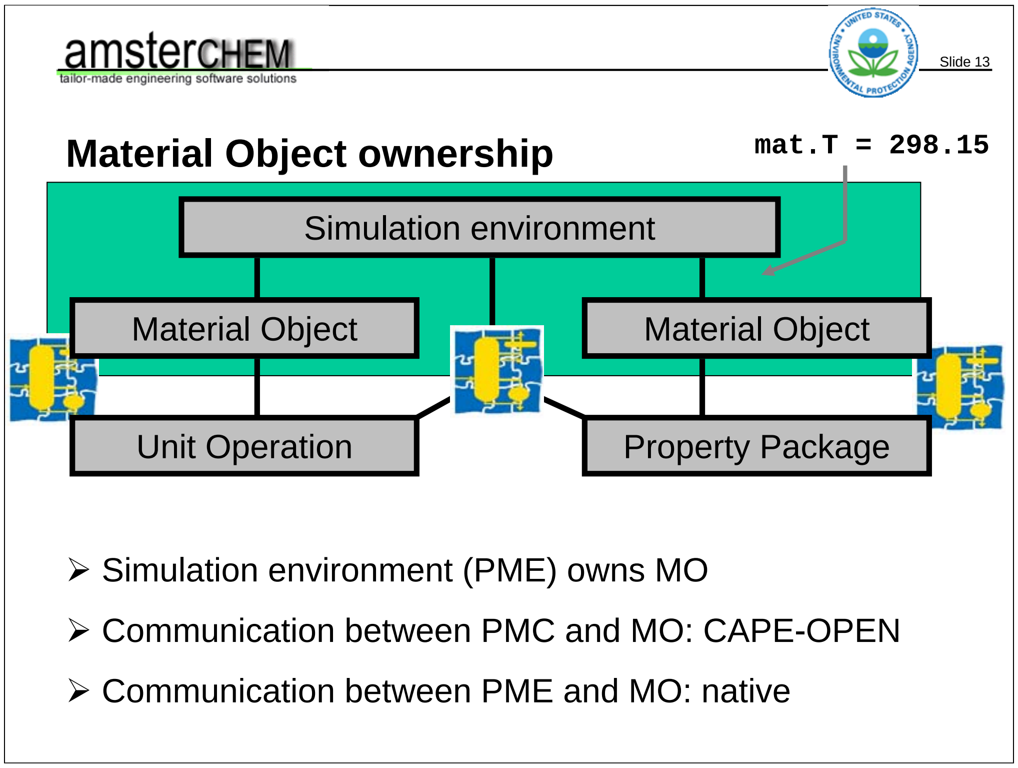
Material Object ownership



- Simulation environment (PME) owns MO
- Communication between PMC and MO: CAPE-OPEN
- Communication between PME and MO: native

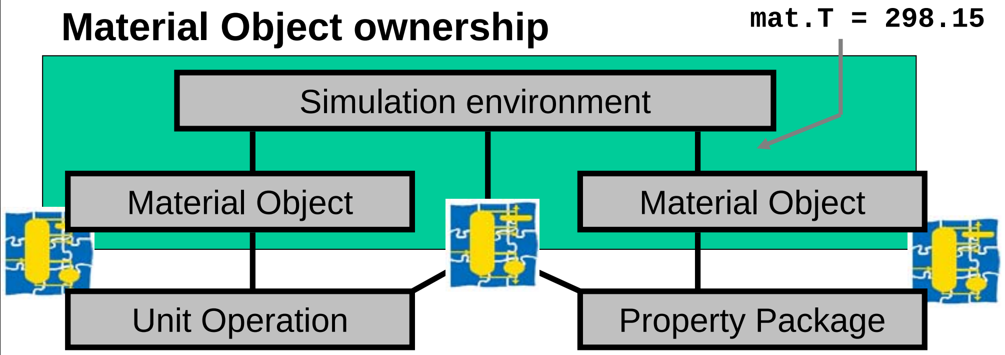
Here we see how it is organized in terms of software components. The Unit Operation PMCs and Property Package PMCs are external components, typically – but not necessarily – implemented by a DLL as COM object. The Material Object is implemented by the simulation application. The green area shows the application space of the simulation application.

So the simulation application communicates with the Unit Operation and with the Property Package. It does so in two ways: directly and via Material Objects. A direct communication with a unit operation for example would be asking the unit operation to perform a calculation, or asking for a collection of the ports on the unit operation. All stream inputs and outputs however are communicated through Material Objects. A similar story holds for the Property Package. A direct request from the simulation application would be a request to perform a calculation. Calculation in- and outputs are communicated through the Material Object.
All interactions between the simulation application and the PMC go via CAPE-OPEN interfaces. The simulation application also needs to set and get data to its Material Objects. These Material Objects are however owned and implemented by the simulation application. Of course the simulation application can make CAPE-OPEN calls on the material objects, but it does not have to. It can access the variables and functions of the Material Object directly. Native communication between a simulation environment and its Material Objects is generally easier to implement, and more-over, much more efficient. …


amsterCHEM
tailor-made engineering software solutions

Slide 13

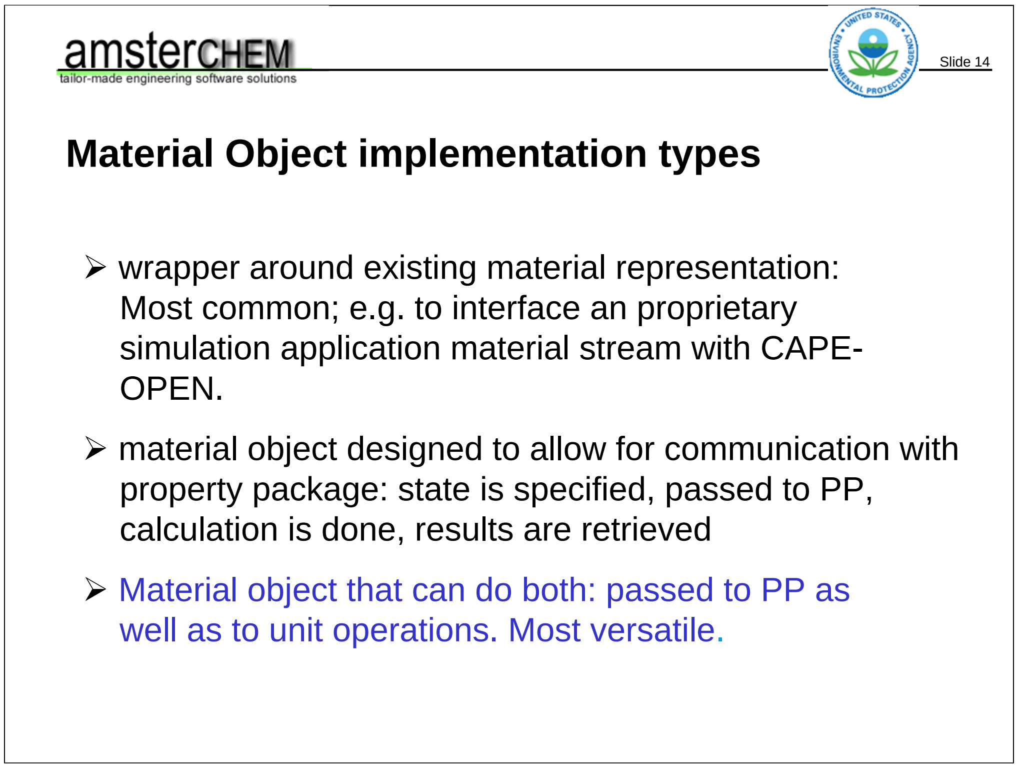
Material Object ownership



- ➢ Simulation environment (PME) owns MO
- ➢ Communication between PMC and MO: CAPE-OPEN
- ➢ Communication between PME and MO: native

… For example, a simulation application could set the overall temperature on a Material Object using the CAPE-OPEN call SetOverall Property, passing the property name – “Temperature” – a place holder for the property basis – nothing – and a VARIANT object containing a double precision array with one element holding the value of temperature. The material would look at the property argument, compare that to the string “Temperature”, check the basis argument, check the type and dimension of the VARIANT object, extract the value from the VARIANT, and set is temperature.

It is all much easier if the simulation application would just do mat.T = value, which it can.


amsterCHEM
tailor-made engineering software solutions

Slide 14

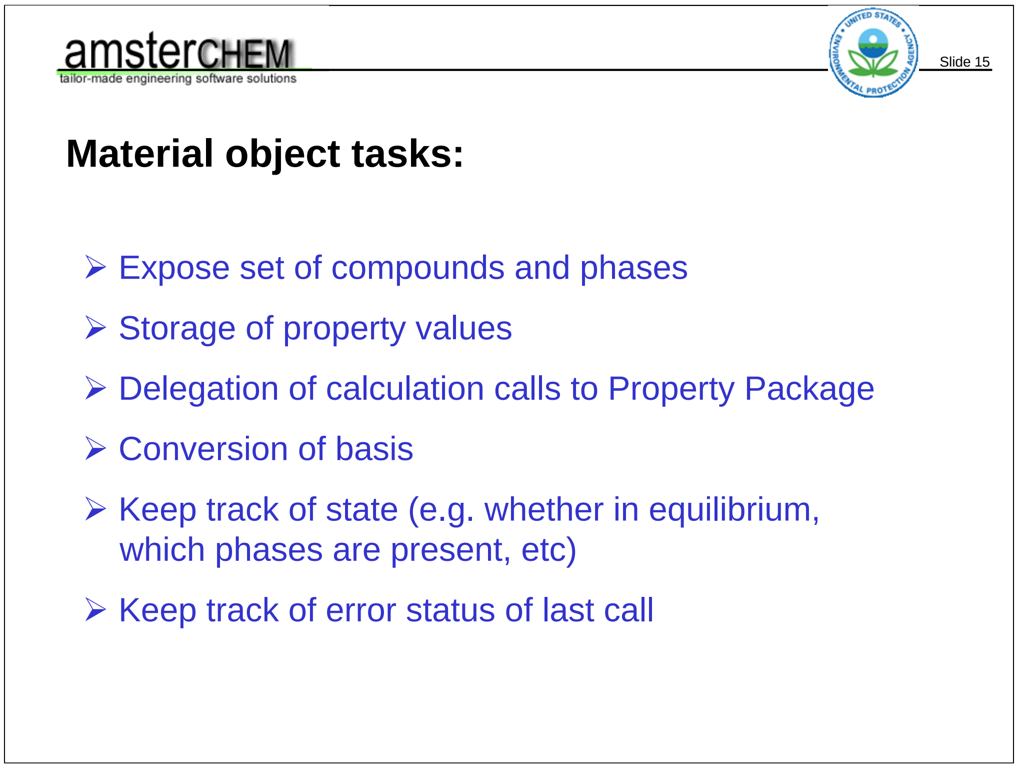
Material Object implementation types

- ➢ wrapper around existing material representation: Most common; e.g. to interface an proprietary simulation application material stream with CAPE-OPEN.
- ➢ material object designed to allow for communication with property package: state is specified, passed to PP, calculation is done, results are retrieved
- ➢ Material object that can do both: passed to PP as well as to unit operations. Most versatile.

Depending on the purpose, we may identify how we want to go about implementing the Material Object. A common situation is that we are dealing with an existing simulation application, and we want to extend its functionality to include CAPE-OPEN objects. This means there is some sort of description of a material or stream already present in the application, and the Material Object we write will be a wrapper between its data and methods and CAPE-OPEN.

Another situation is that we want to have a Material Object only to be able to perform thermodynamic calculations using a CAPE-OPEN property package. These types of Material Objects can be rather simple, as we know in advance exactly which property we expect.

A more general type of Material Object we will concentrate on today: one that can be passed to Unit Operations as well as Property Packages.

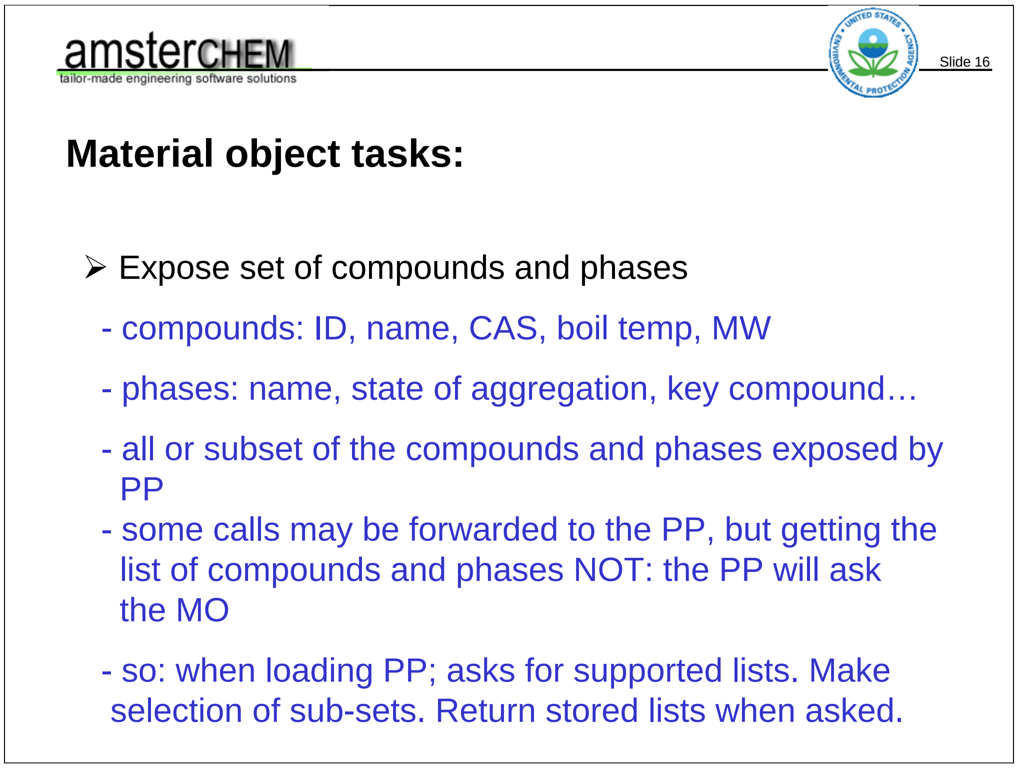

amsterCHEM
tailor-made engineering software solutions

Slide 15

Material object tasks:

- Expose set of compounds and phases
- Storage of property values
- Delegation of calculation calls to Property Package
- Conversion of basis
- Keep track of state (e.g. whether in equilibrium, which phases are present, etc)
- Keep track of error status of last call

To be able to work with Material Objects, a Material Object must implement a number of tasks… (task enumeration) In the next slides, each of these tasks will be elaborated upon.


amsterCHEM
tailor-made engineering software solutions

Slide 16

Material object tasks:

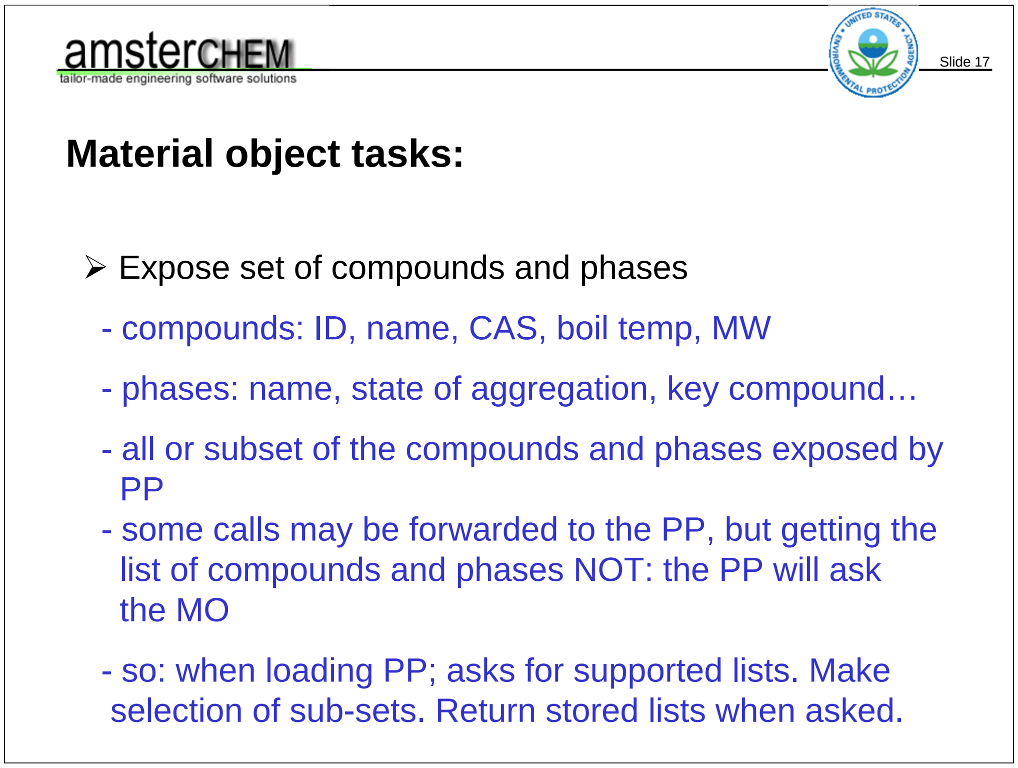
- Expose set of compounds and phases
  - compounds: ID, name, CAS, boil temp, MW
  - phases: name, state of aggregation, key compound…
  - all or subset of the compounds and phases exposed by PP
  - some calls may be forwarded to the PP, but getting the list of compounds and phases NOT: the PP will ask the MO
  - so: when loading PP; asks for supported lists. Make selection of sub-sets. Return stored lists when asked.

The Material Object must expose which compounds and phases are present and supported. For compounds, these lists are the same. The material object will typically support a list of compounds. All these compounds are both supported and present. Possibly with a composition (fraction) of zero.
For phases these lists are different. A material object may support phases “Vapor”, “Liquid1” and “Liquid2”, but at any given time, the phases present (and possibly in equilibrium) may be only “Vapor”, or for example only “Liquid1” and “Liquid2”.

The information about compounds that are present and supported is typically the compound ID, name, CAS registry number, normal boiling point and molecular weight. The compound ID is the most important one; this string value must be unique for each compound, and is used in the communication with a Material Object and PME to identify compounds. It may very well have the same value as the compound name.

Phases are treated differently in version 1.0 and version 1.1 thermo. In version 1.0 thermo, originally only phases “Vapor”, “Liquid” and “Solid” were allowed. This was later changed into allowing multiple phases of the same aggregation state, as long as the phase ID started with the aggregation state. For example, a version 1.0 Property Package or Material Object could support liquid phases “Liquid” and “Liquid1” or “LiquidWater”.

If the thermodynamic calculations are performed by a Property Package, typically the Material Object exposes the same set as compounds and phases as exposed by the Property Package. This is not required though. A property package may support 15 different compounds, only 10 of which are actually used in the simulation. The Property Package would then expose all 15 supported compounds, whereas the material object only supports the 10 compound used in the simulation. A Unit Operation talking to the Material Object would only see the 10 exposed


amsterCHEM
tailor-made engineering software solutions

Slide 17

Material object tasks:

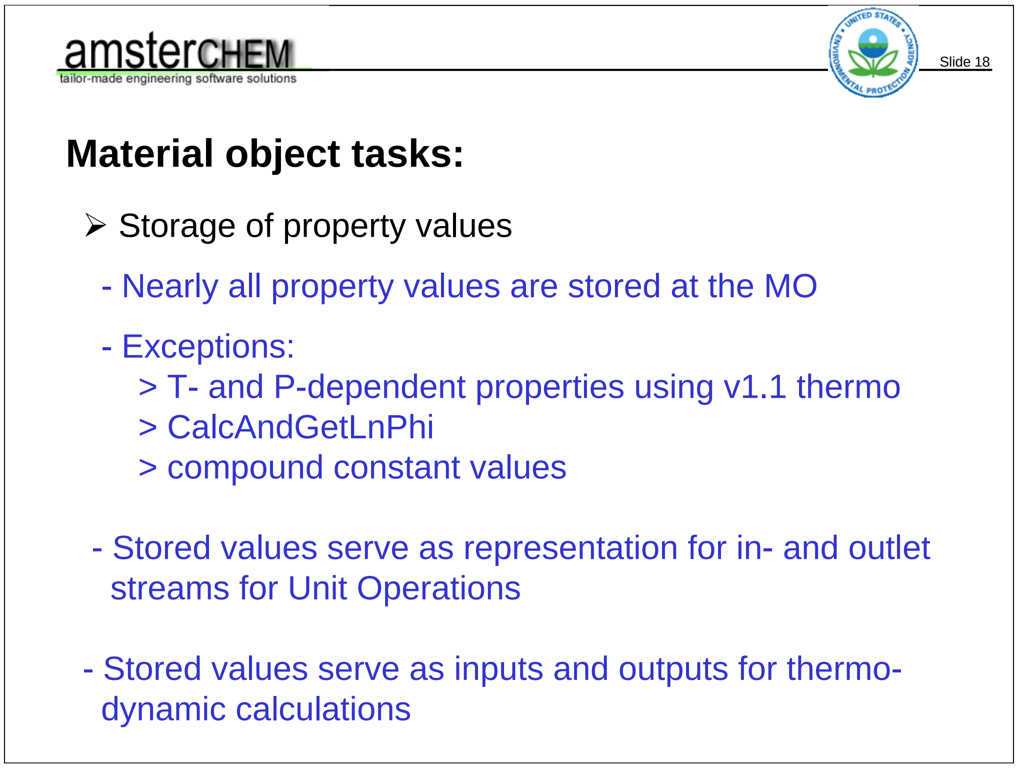
- Expose set of compounds and phases

  - compounds: ID, name, CAS, boil temp, MW
  - phases: name, state of aggregation, key compound…
  - all or subset of the compounds and phases exposed by PP
  - some calls may be forwarded to the PP, but getting the list of compounds and phases NOT: the PP will ask the MO
  - so: when loading PP; asks for supported lists. Make selection of sub-sets. Return stored lists when asked.

…

The Material Object must expose which compounds and phases are present and supported. For compounds, these lists are the same. The material object will typically support a list of compounds. All these compounds are both supported and present. Possibly with a composition If a Property Package is used to do the calculations, the Material Object will typically forward many calls that are made by PMCs – for example a unit operation – directly to the Property Package. This must not be done for the functions that expose compounds and phases though. For a vapor enthalpy calculation performed by the Property Package, the Property Package will start by asking: for which compounds does the calculation need to be performed? It will ask the material. If the material asks the Property Package in turn, you run the risk of an infinite loop (in this case not likely, because the Property Package will not ask the Material Object when it gets a request for which compounds it supports).

In either case, the calls for supported compound and phase lists are made rather frequently. It is simply not efficient to forward these calls each time. The thing to do is – when you load the Property Package – you ask for the supported compounds and phases once. Then you may make a selection of sub sets of these. The simulation application remembers the resulting lists, and these lists are directly returned by the Material Object.



amsterCHEM
tailor-made engineering software solutions

Slide 18

Material object tasks:

- Storage of property values

  - Nearly all property values are stored at the MO

  - Exceptions:
    > T- and P-dependent properties using v1.1 thermo
    > CalcAndGetLnPhi
    > compound constant values

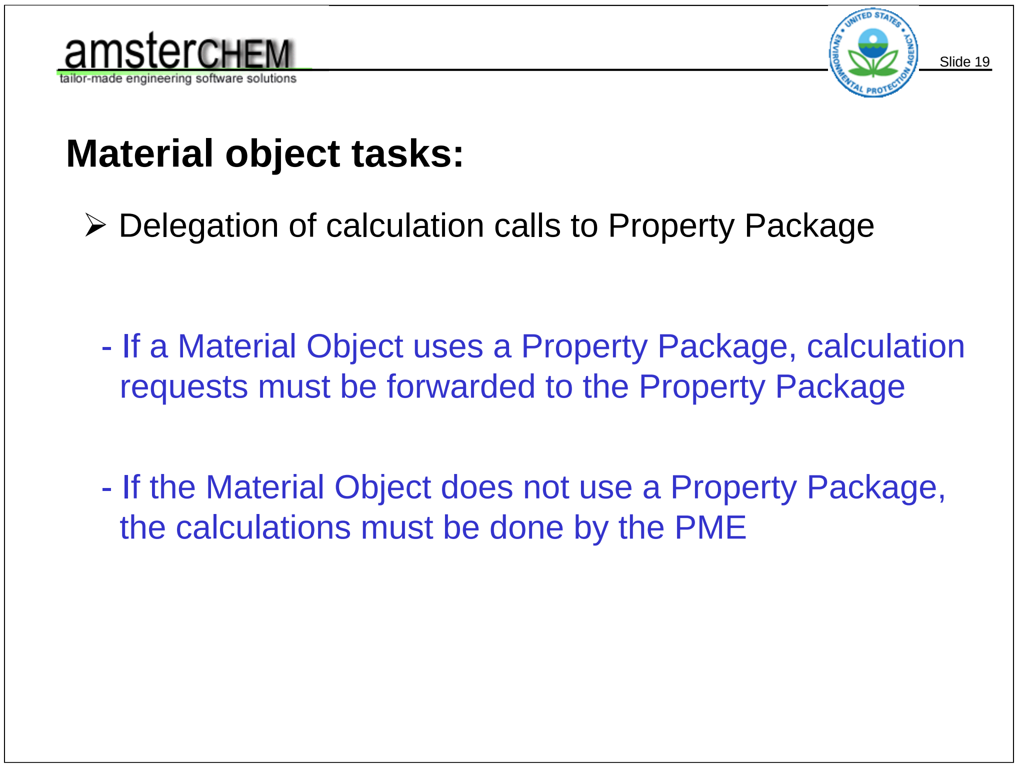
  - Stored values serve as representation for in- and outlet streams for Unit Operations

  - Stored values serve as inputs and outputs for thermo-dynamic calculations

The main task of the Material Object is to store thermodynamic and physical properties. This functionality will be implemented by CAPE-OPEN interfaces ICapeThermoMaterialObject (version 1.0 thermo) or ICapeThermoMaterial (version 1.1 thermo). Nearly all thermodynamic and physical properties that are communicated between PMCs and the simulation application are stored at the Material Object. There are a few exceptions. When using version 1.1 thermo, calculation of compound properties that depend only on pressure or temperature are not stored at the Material Object, they are returned directly as a result of the function call. Similarly, the version 1.1 thermo interface exposes a shortcut method for obtaining results of fugacity calculations. The calculation results are returned as arguments to the function call.

Also values that are constant for each compound – such as critical properties, molecular weight, … - are not stored, but directly retrieved by the function call. This is the case for both version 1.0 and version 1.1 thermo.

The properties that are stored on the material object can represent the state of a stream in the flowsheet. As such, they serve as inputs and outputs for Unit Operations. As shown before, the stored properties also represent the in- and outputs to thermodynamic calculation. If a Unit Operation would want to know vapor enthalpy, it would store pressure, temperature and composition on the Material Object, then ask the Material Object to do the calculation of vapor enthalpy. If the Material Object uses a Property Package to do its calculations, it will forward this call to the Property Package. The calculation is done, and stored on the Material Object. Control is returned to the Unit Operation, that will then retrieve the calculated vapor enthalpy from the Material Object.


amsterCHEM
tailor-made engineering software solutions

Slide 19

Material object tasks:

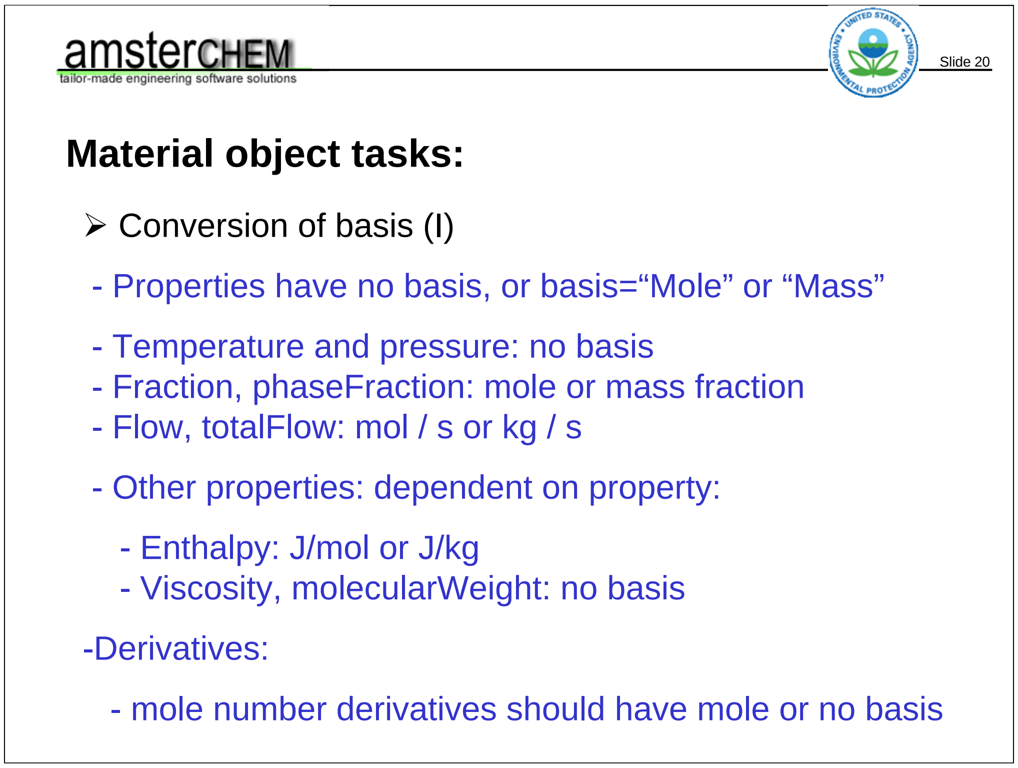
- ➢ Delegation of calculation calls to Property Package

  - If a Material Object uses a Property Package, calculation requests must be forwarded to the Property Package

  - If the Material Object does not use a Property Package, the calculations must be done by the PME

In the previous example, the unit operation asked the Material Object to calculate the vapor enthalpy. The Material Object in turn asks the Property Package to do the calculation. The Unit Operation never talks directly to the Property Package. In fact, there might not even be a a Property Package that is used for thermodynamic calculations. The Unit Operation will never know. It just asks the Material Object to do any calculation.

If there is no Property Package, the requested calculation must be performed by the simulation application somehow. This is of course implementation dependent.


amsterCHEM
tailor-made engineering software solutions

Slide 20

Material object tasks:

➢ Conversion of basis (I)

- Properties have no basis, or basis=“Mole” or “Mass”

- Temperature and pressure: no basis
- Fraction, phaseFraction: mole or mass fraction
- Flow, totalFlow: mol / s or kg / s

- Other properties: dependent on property:

  - Enthalpy: J/mol or J/kg
  - Viscosity, molecularWeight: no basis

-Derivatives:

  - mole number derivatives should have mole or no basis

Conversion of basis is a task of the Material Object. It is one of the more complicated tasks when implementing a Material Object. So let’s have a closer look at the basis concept.
A property either has no basis, or it has a basis. If it has a basis, the basis is molar or mass. Basis is identified by the string “mole” or “mass”, or an empty string (NULL in COM)

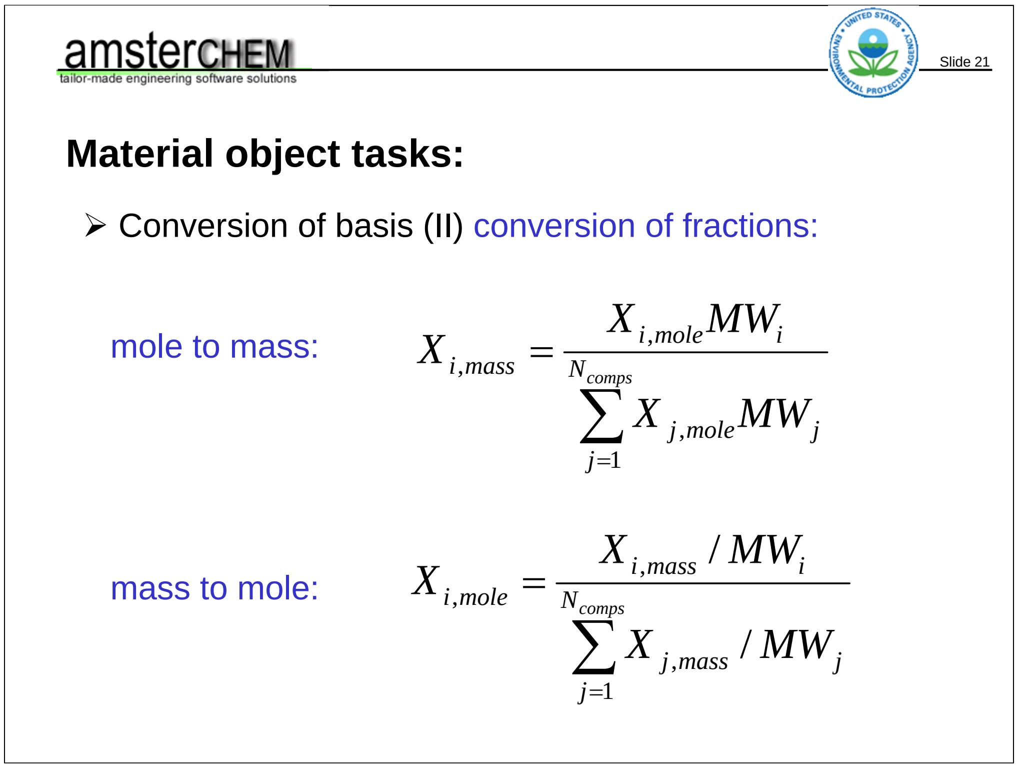
Of the special properties, temperature and pressure have no basis. Their units or measure are Kelvin and Pascal. Basis should not be confused with unit of measure.

Fraction and phase fraction have a basis. Depending on the basis, they are represented as mole or mass fractions.

Flow and totalFlow (remember that flow is the compound flow) have a basis. Their units of measure are mol / s or kg / s, depending on the basis.

Other properties may or may not have a basis. This depends no the property. Typically, the ones having a basis are the extensive ones, such as enthalpy. Its unit of measure is J/mol or J/kg, depending on the basis. With a few exceptions (such as density) intensive properties do not have a basis. An example is viscosity. A special case is property molecularWeight. It is relative w.r.t. the mass of carbon divided by 12, hence dimensionless, has no unit of measure. If you want to give it a unit of measure, call it gr/mol. It has no basis.

Temperature and pressure derivatives change the unit of measure of a property, but not its basis. Mole fraction derivatives change neither the basis nor the unit of measure. Mole number derivatives are a special case. It is advised to not use the mass basis for mole number derivatives. More information on this is given in the version 1.1 spec.



amsterCHEM
tailor-made engineering software solutions

Slide 21

Material object tasks:

➢ Conversion of basis (II) conversion of fractions:

mole to mass:

$$X_{i,mass} = \frac{X_{i,mole} MW_i}{\sum_{j=1}^{N_{comps}} X_{j,mole} MW_j}$$

mass to mole:

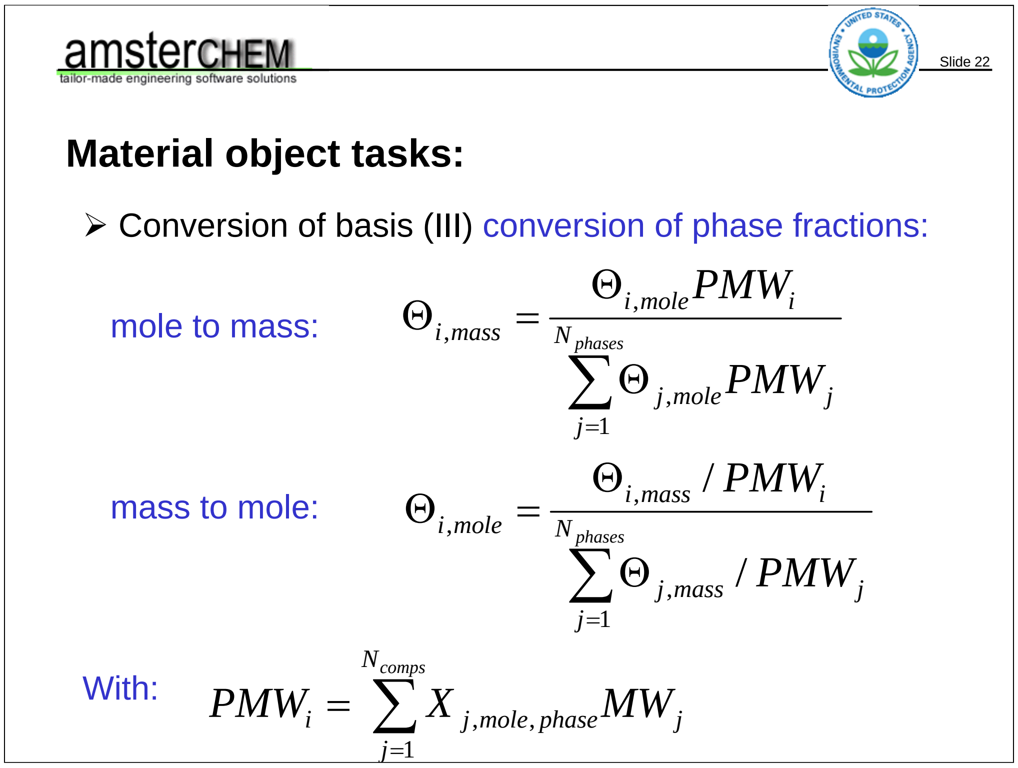
$$X_{i,mole} = \frac{X_{i,mass} / MW_i}{\sum_{j=1}^{N_{comps}} X_{j,mass} / MW_j}$$

Conversion of fractions – composition – from mole to mass basis is a matter of multiplying with the molecular weight and renormalizing. Hence, this conversion cannot be done – as with all other mass / mole conversions – if not all molecular weights are known. This can be the case, for example when your simulation involves a compound like cement.

The composition conversion can also not be done if the composition is only partially known, as it involves renormalizaton. In other words, in a system with methane, ethane and propane, you cannot convert a mole fraction of ethane to a mass fraction if the compositions of ethane and propane are not known.

Conversion of mass fraction to mole fraction is similar, but divide rather than multiple by molecular weight.

Note that this presumes the starting fractions add up to unity. If we start with a total mole fraction of 0.9, these formulas will give us a total mass fraction of 1.0. Of course starting with a total mole fraction of 1.0 this answer is unique and correct. If not, the answer is subject to choice, and it is debatable what is correct. To end up with a total mass fraction of 0.9 does not appear to be correct, hence we make the normalization choice here.


amsterCHEM
tailor-made engineering software solutions

Slide 22

Material object tasks:

- Conversion of basis (III) conversion of phase fractions:

mole to mass:

$$\Theta_{i,mass} = \frac{\Theta_{i,mole} PMW_i}{\sum_{j=1}^{N_{phases}} \Theta_{j,mole} PMW_j}$$

mass to mole:

$$\Theta_{i,mole} = \frac{\Theta_{i,mass} / PMW_i}{\sum_{j=1}^{N_{phases}} \Theta_{j,mass} / PMW_j}$$

With:

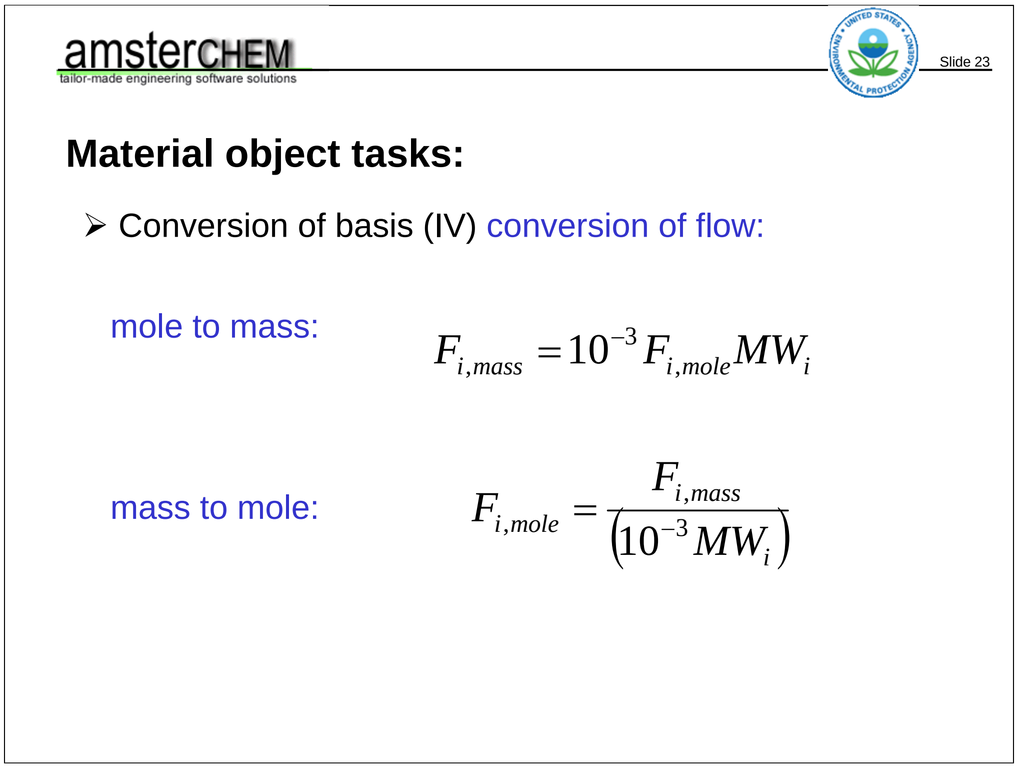
$$PMW_i = \sum_{j=1}^{N_{comps}} X_{j,mole,phase} MW_j$$

Mass / mole conversion of phase fractions is a bit more complex. The formula is similar to conversion of compositions, but rather than the fractions, the summation and normalization is over all phase fractions, and rather than the compound molecular weight you use the phase molecular weight PMW. The phase molecular weight can be calculated as the summation over composition in mole fractions in that phase times the compound molecular weight.

Hence, to do a mole / mass conversion for phase fractions, all phase fractions and all phase compositions must be known.

Notice that this formula assumes that all phase fractions are known in the same basis, which is likely but not necessarily the case. If this is not the case, and you still want to support the conversion, the calculation is only slightly more complex.

Also shown here is the calculation for the phase molecular weight presuming the phase compositions are known in mole basis. One can of course simply apply the fraction conversion formula on the previous slide if the phase composition is known in mass basis.



amsterCHEM
tailor-made engineering software solutions

Material object tasks:

- Conversion of basis (IV) conversion of flow:

mole to mass:

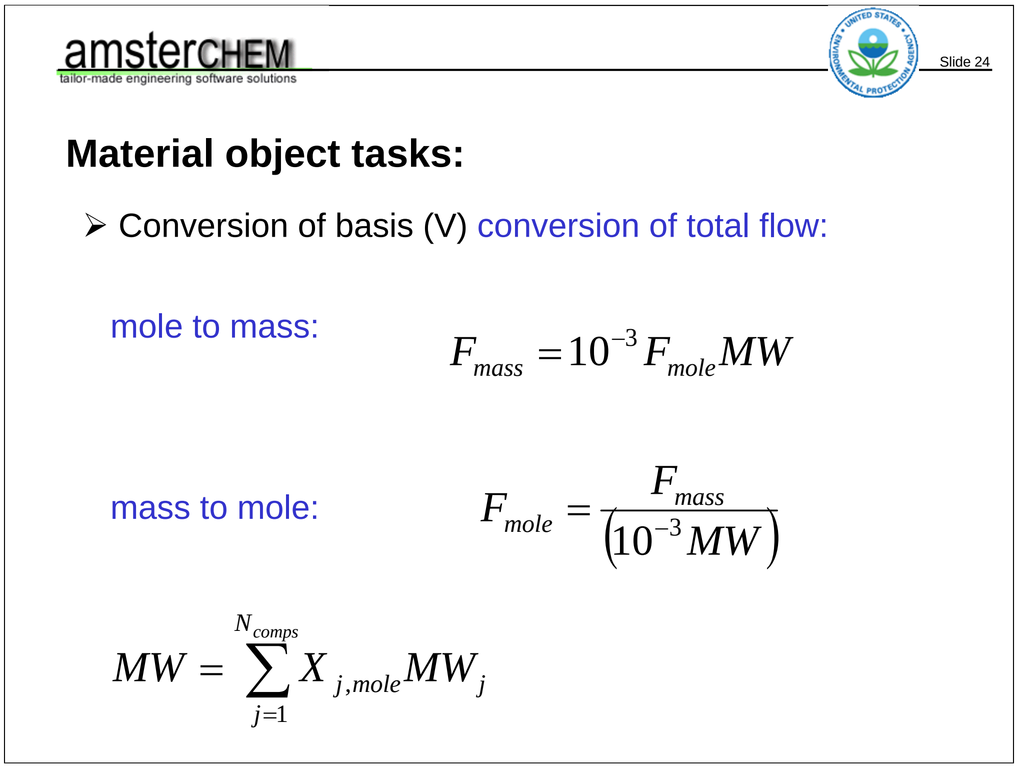
$$F_{i,mass} = 10^{-3} F_{i,mole} MW_i$$

mass to mole:

$$F_{i,mole} = \frac{F_{i,mass}}{\left(10^{-3} MW_i\right)}$$

Conversion of compound flow for compound i from mole to mass basis is a matter of multiplication by its molecular weight. Do not forget the correction factor, as we have relative molecular weight, instead of kg/mol.

Mass to mole: divide by the corrected molecular weight. Here, the molecular weight for conversion of compound i is that of compound i.


amsterCHEM
tailor-made engineering software solutions

Slide 24

Material object tasks:

- Conversion of basis (V) conversion of total flow:

mole to mass:

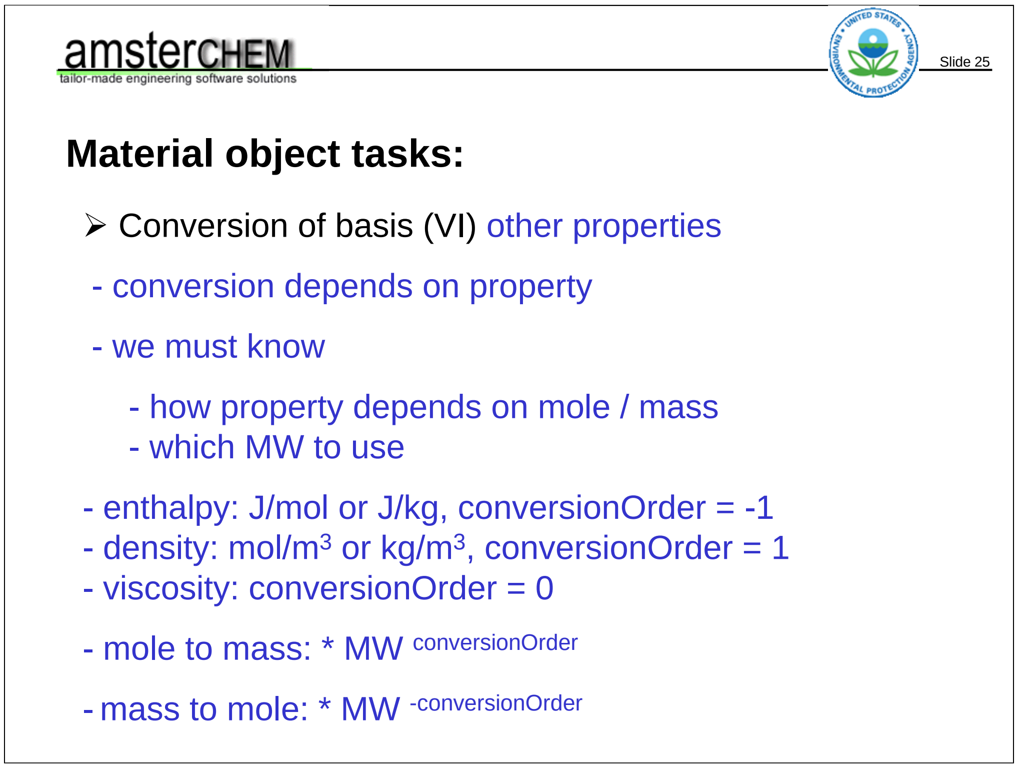
$$F_{mass} = 10^{-3} F_{mole} MW$$

mass to mole:

$$F_{mole} = \frac{F_{mass}}{\left(10^{-3} MW\right)}$$

$$MW = \sum_{j=1}^{N_{comps}} X_{j,mole} MW_j$$

Conversion for total flow is very similar to the conversion of compound flow. Rather than the compound molecular weight we need to use the mixture molecular weight in this case. This happens to be the same as the phase molecular weight we saw earlier: the summation of molar composition times molecular weight of each compound.



amsterCHEM
tailor-made engineering software solutions

Slide 25

Material object tasks:

- Conversion of basis (VI) other properties

  - conversion depends on property

  - we must know

    - how property depends on mole / mass
    - which MW to use

  - enthalpy: J/mol or J/kg, conversionOrder = -1
  - density: mol/m^3 or kg/m^3, conversionOrder = 1
  - viscosity: conversionOrder = 0

  - mole to mass: * MW $^{\text{conversionOrder}}$

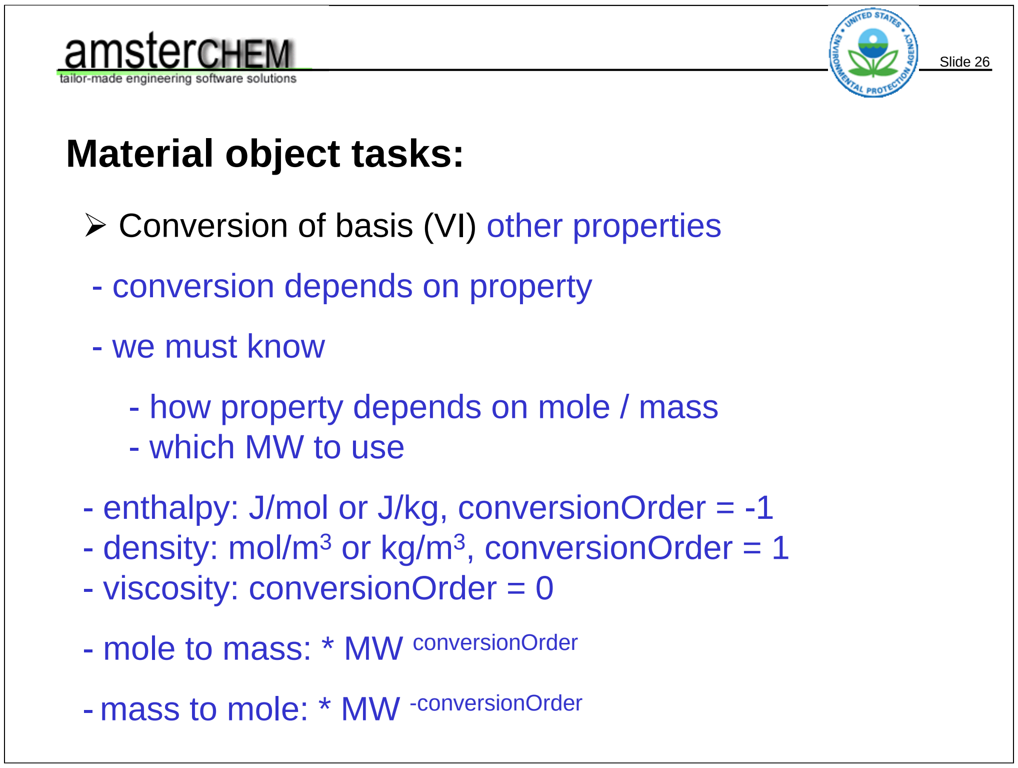
  - mass to mole: * MW $^{\text{-conversionOrder}}$

Basis conversions of other properties depends on the properties at hand. For implementation purposes, it is convenient to generalise the approach. We should essentially know – for each property – how it depends on mass or mole. This can of course be derived from it unit of measure. Also, we should know which molecular weight to use for the basis conversions. This can be the compound molecular weight, or the phase mixture molecular weight that we have defined earlier. The phase molecular weight and compound molecular weight are of course equal for systems with a single compound. For systems with multiple compounds, the composition is required to calculate the phase molecular weight.

Let us start by generalizing our conversion approach by defining a conversion order. The conversion order is the power of mole or mass in the unit of measure. For all extensive properties – like enthalpy – it has a value of -1. For all properties that do not depend on basis, it has a value of 0; basis conversions do not apply to these properties. With one exception, -1 and 0 are the only conversionOrders you will find for properties that are currently defined in the scope of CAPE-OPEN. The one exception is density: it has a conversion order of 1.

To go from mole to mass basis, we multiply with molecular weight to the power of conversionOrder. In the other direction, the conversion order simply changes sign.

…


amsterCHEM
tailor-made engineering software solutions

Slide 26

Material object tasks:

- Conversion of basis (VI) other properties

 - conversion depends on property

 - we must know

   - how property depends on mole / mass
   - which MW to use

- enthalpy: J/mol or J/kg, conversionOrder = -1
- density: mol/m^3 or kg/m^3, conversionOrder = 1
- viscosity: conversionOrder = 0

- mole to mass: * MW $^{\text{conversionOrder}}$

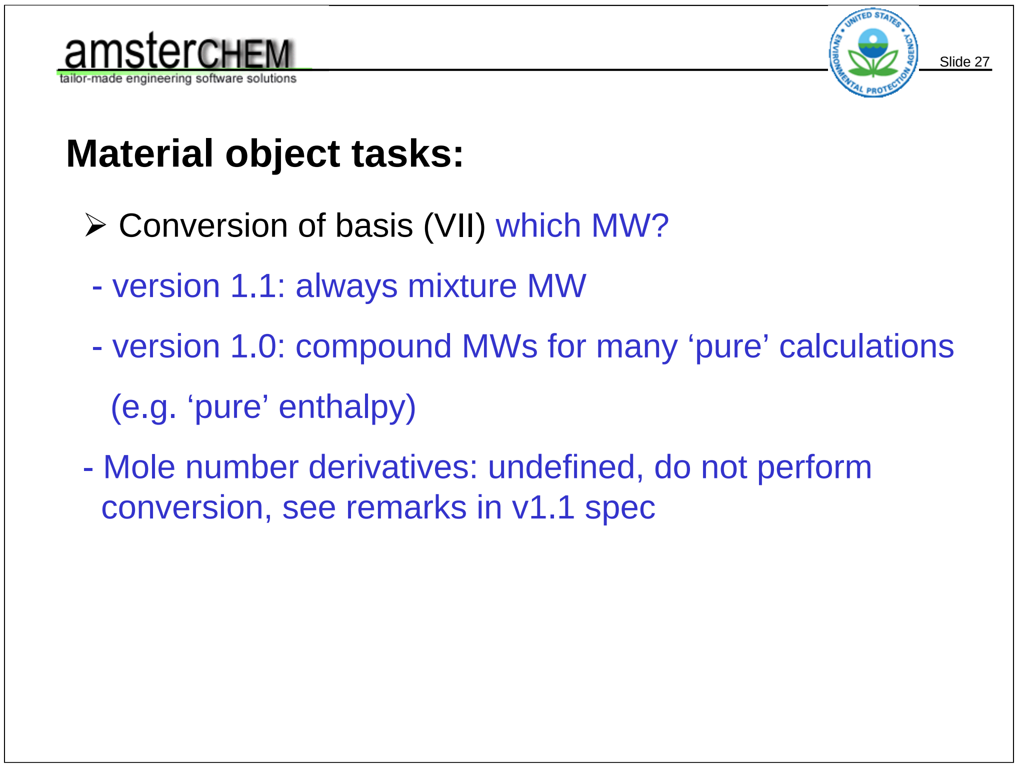
- mass to mole: * MW $^{\text{-conversionOrder}}$

…

So our algorithm could be: get the properties conversion order. If not zero, check if the property is known in the same basis as it was stored. In that case, set conversionOrder to zero, as no conversion is required. If not zero, check if it is mass to mole basis conversion. In that case, set the conversion order to its negative value. Now we end up with a number that – if nonzero – is the power for the molecular weight in the conversion.

As the power can only be 1 or -1 with the current property definitions, actually using a power calculations is of course not very efficient. Best to check for 1, in which case the result must be multiplied by molecular weight, or -1, in which case the result must be divided by molecular weight.

Don't forget to correct molecular weight by a factor 1/1000 because of the kg/mol rather than gr/mol units required for conversion.


amsterCHEM
tailor-made engineering software solutions

Slide 27

Material object tasks:

- Conversion of basis (VII) which MW?

 - version 1.1: always mixture MW

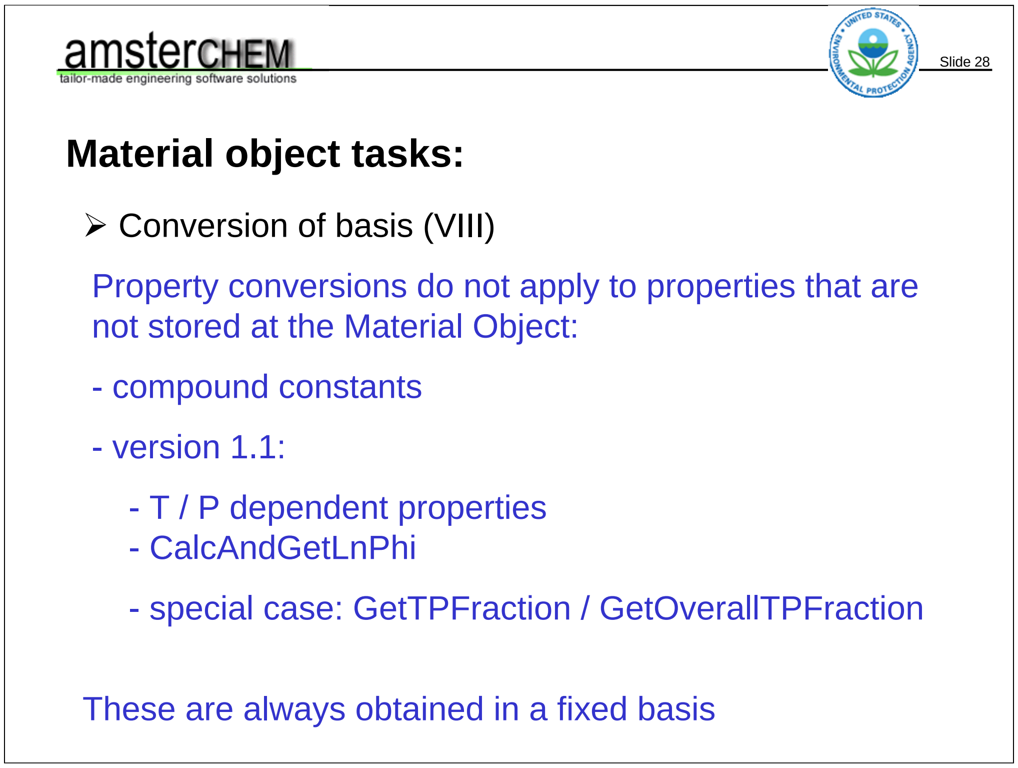
 - version 1.0: compound MWs for many 'pure' calculations

 (e.g. 'pure' enthalpy)

- Mole number derivatives: undefined, do not perform conversion, see remarks in v1.1 spec

This leaves us the question which molecular weight to use. Generally, this is the mixture molecular weight. The story is somewhat more complex for version 1.0, where we can specify 'pure' property calculations. For mixture properties such as enthalpy, that are calculated using the 'pure' calculation type. For any 'pure' calculations, the v1.0 spec states: "pure" means that the property values refer to the components when they are in pure state (not mixed with other components). This means that composition is not actually used for these calculations, and therefore should also not affect the basis conversion. The compound molecular weights should be used here.

Pure calculations are not frequently used for mixture properties. It is perfectly ok not to support them.


amsterCHEM
tailor-made engineering software solutions

Slide 28

Material object tasks:

➢ Conversion of basis (VIII)

Property conversions do not apply to properties that are not stored at the Material Object:

- compound constants
- version 1.1:
    - T / P dependent properties
    - CalcAndGetLnPhi
    - special case: GetTPFraction / GetOverallTPFraction

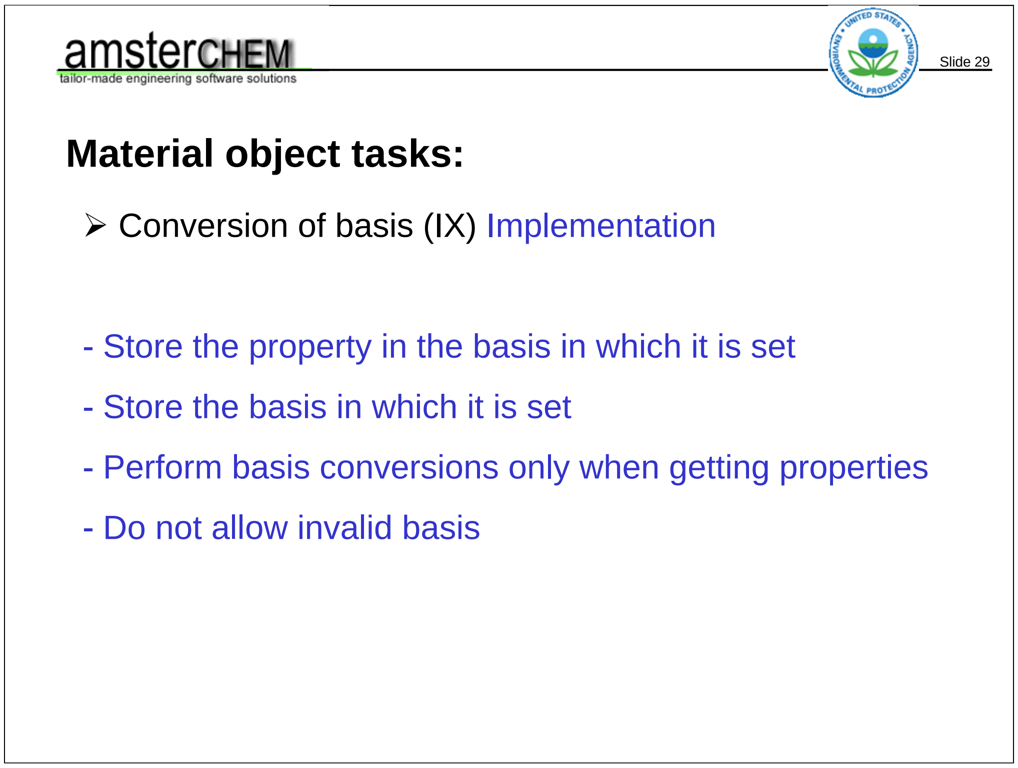
These are always obtained in a fixed basis

Property conversions are performed by the Material Object. They therefore do not apply to properties that are not stored at the Material Object.

Compound constants for example always have a fixed basis – generally mole basis. The unit of measure for liquidDensityAt25C is always mol/m3, never kg/m3.

The same holds for pressure and temperature dependent properties when using version 1.1 thermo. More on these properties later.

Version 1.1 has some shortcut functions. One for calculation of fugacities. This takes composition always in mole fraction. Another shortcut is GetTPFraction. Again, composition is always in mole fraction.


amsterCHEM
tailor-made engineering software solutions

Slide 29

Material object tasks:

- Conversion of basis (IX) Implementation

- Store the property in the basis in which it is set
- Store the basis in which it is set
- Perform basis conversions only when getting properties
- Do not allow invalid basis

A material object is demanded to always be able to return a property in the basis in which it was set. This is a very useful feature in simulations in which basis conversions cannot be done, in the case molecular weights are not known for example. Or, if composition is not known. Hence, we should store the property always in the basis in which it was set. This means, we also need to remember for each property in which basis it was set.

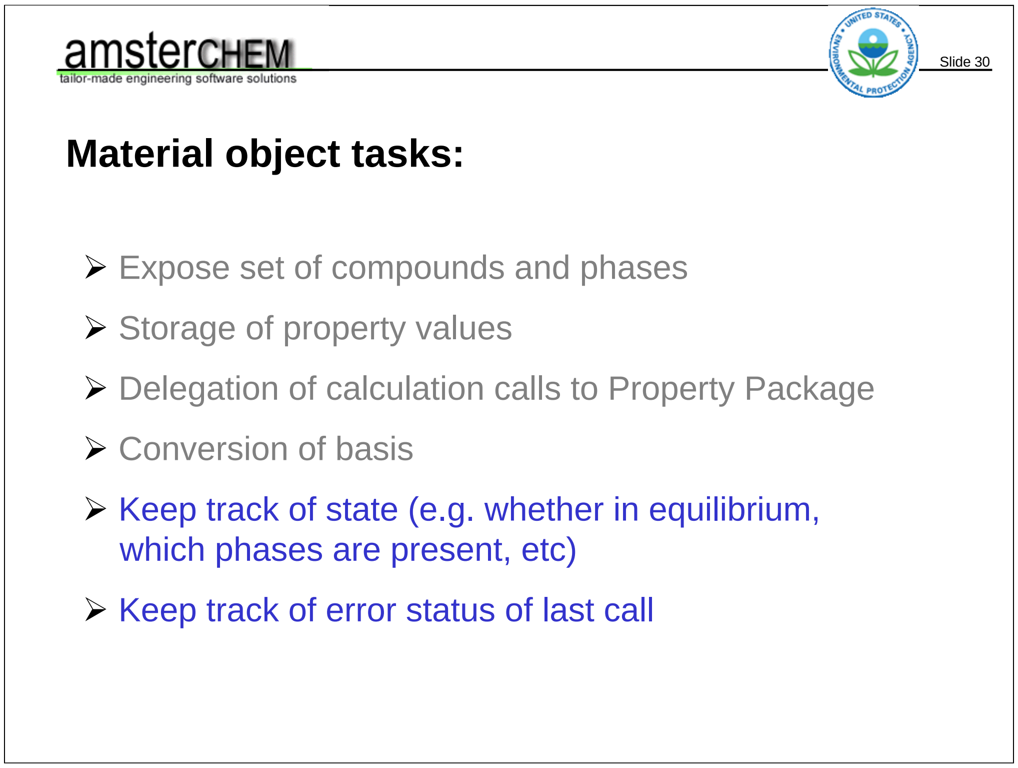
Then, if the property is retrieved from the material object, we apply basis conversion if required, using the scheme as outlined in previous sheets.

Storing properties always in the same basis, for example always in mole basis, is a bad idea. For starters, if the property was set in mass basis, and obtained in mass basis, we are converting back and forth to mole basis for no reason. Not very efficient.

More importantly, consider the following order of operations: a unit operation specifies enthalpy in mass basis. Next, it specifies composition.

At the moment enthalpy was stored in mass basis, if we would have converted to mole basis, we require the composition. This was not yet set by the unit operation. So it may be unknown, or present at the material from a previous operation. The former would result an error, the latter would result something worse: no error, but wrong results.

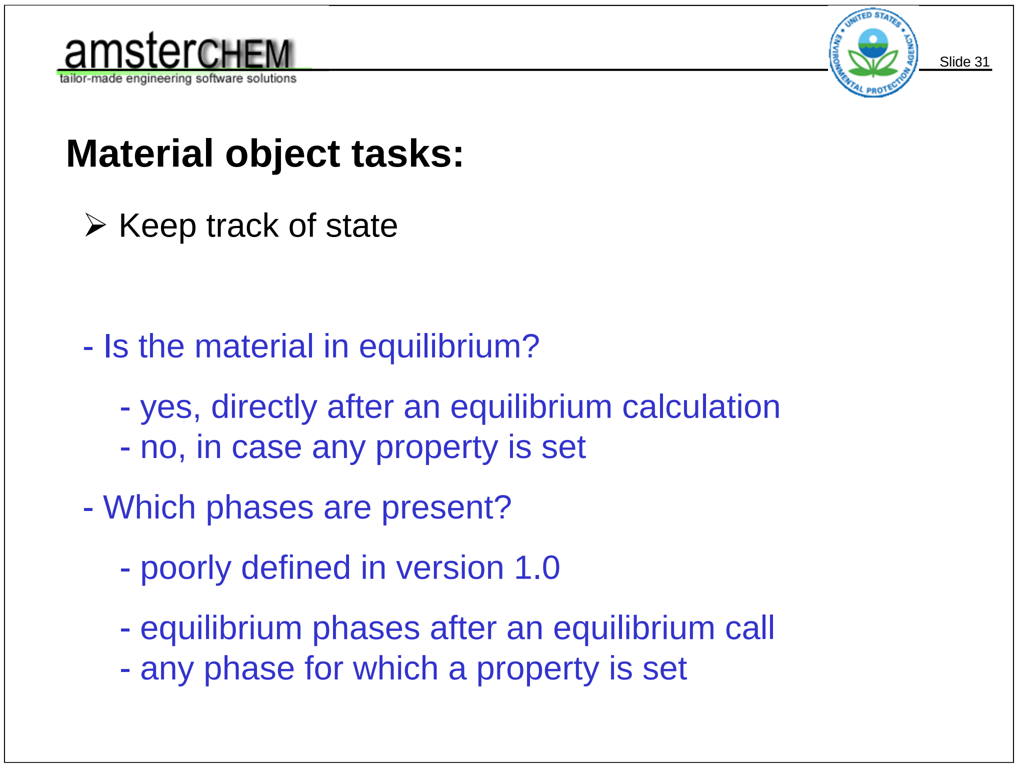
Final remark: always check the basis for setting and getting properties. The material object should not allow for setting temperature in mole basis, nor should it allow for getting enthalpy without a basis (in the past at some point getting enthalpy without a basis had a different meaning: the total enthalpy, e.g. J or J/s; due to ambiguous documentation, one should not support this).


amsterCHEM
tailor-made engineering software solutions

Material object tasks:

- Expose set of compounds and phases
- Storage of property values
- Delegation of calculation calls to Property Package
- Conversion of basis
- Keep track of state (e.g. whether in equilibrium, which phases are present, etc)
- Keep track of error status of last call

Now that we have covered basis conversions, back to the Material Object tasks. Two more tasks to cover…


amsterCHEM
tailor-made engineering software solutions

Slide 31

Material object tasks:

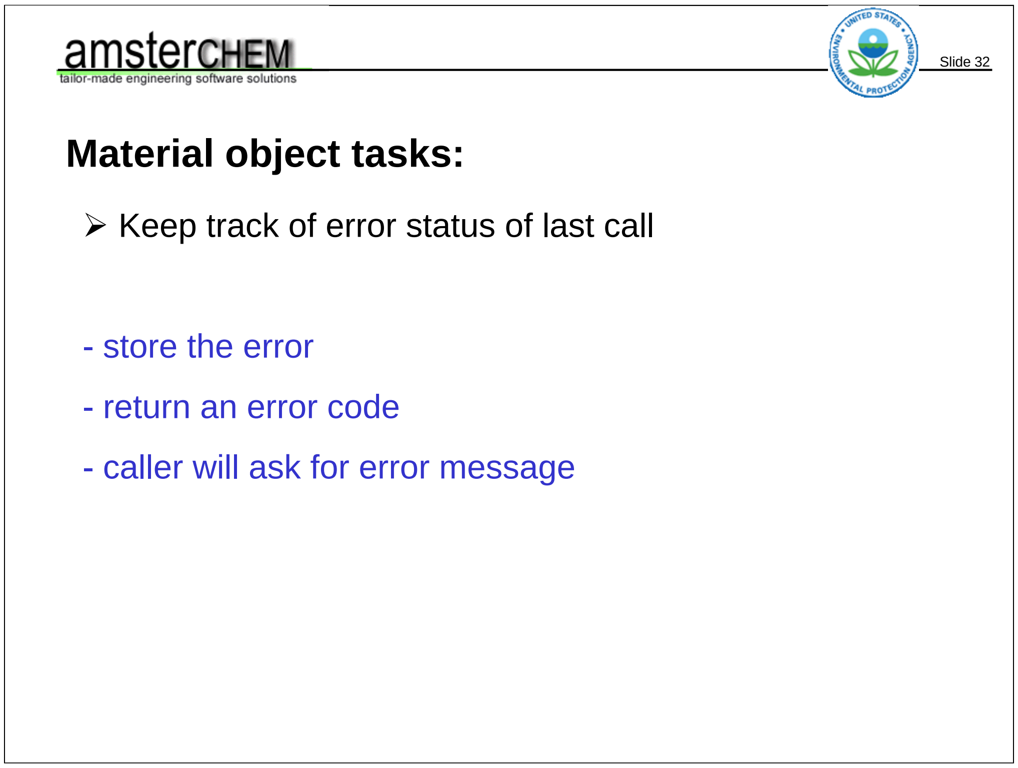
➢ Keep track of state

- Is the material in equilibrium?
  - yes, directly after an equilibrium calculation
  - no, in case any property is set
- Which phases are present?
  - poorly defined in version 1.0
  - equilibrium phases after an equilibrium call
  - any phase for which a property is set

The simulator will want to know at various points whether a Material Object is currently in equilibrium. For example to check whether the inlet streams to a Unit Operation are ready for calculating the unit operation and for post-checking after calculating the unit operation whether the unit operation has done its task of flashing all outlet streams. Also, version 1.1 defines a function of getting all present phases and their status. This function also requires to know whether the Material Object, and hence the phases, are in equilibrium. Keeping track of equilibrium state is easy: initially not. After an equilibrium calculation: yes. If we set any property, then no.

It is also up to the Material Object to keep track of which phases are currently present. A version 1.1 equilibrium calculations allows calculation for an equilibrium on a subset of phases. The procedure for calculating an equilibrium using version 1.1 thermo is: set all phases that are to take part in the calculation as present, calculate the equilibrium, set all resulting equilibrium phases as present.

Generally properties for phases that are not present cannot be obtained from a Material Object. Without having to set the present phases on a Material Object, CAPE-OPEN allows for the following sequence: set pressure, temperature and composition for a given phase, calculate a phase property, obtain the result. For such a sequence, the phase should automatically be marked as present. Hence, we mark a phase as present at the moment any property is set for it.



amsterCHEM
tailor-made engineering software solutions

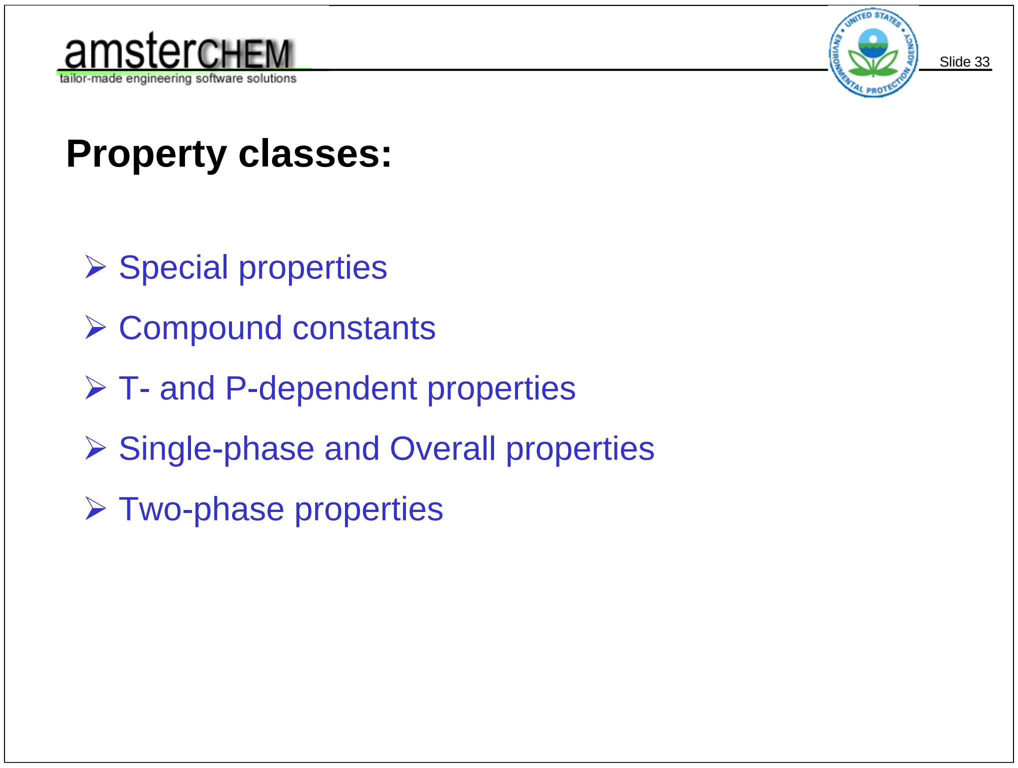
Slide 32

Material object tasks:

- Keep track of error status of last call

- store the error
- return an error code
- caller will ask for error message

Last task: keep track of error data when an error occurs. This task is common for all CAPE-OPEN objects, hence also for Material Objects. In CAPE-OPEN, as soon as an object encounters an error, it returns a CAPE-OPEN error code. This code is just a number, the caller will likely want to know more information about the error, such as a textual description of the error. So before we return the error code, we must remember which information to return – such as the error description – in case the caller asks for it.

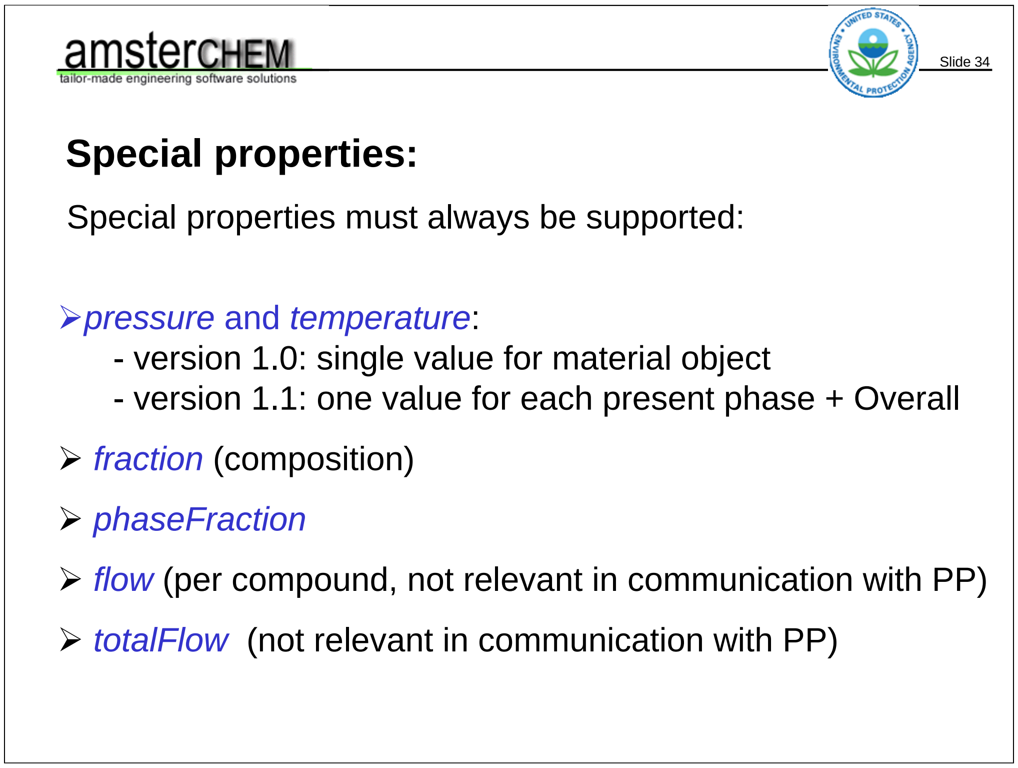

amsterCHEM
tailor-made engineering software solutions

Slide 33

Property classes:

- Special properties
- Compound constants
- T- and P-dependent properties
- Single-phase and Overall properties
- Two-phase properties

The thermodynamic and physical properties defined by CAPE-OPEN can be naturally divided in a number of property classes. This classification is very useful. So useful, that the version 1.1 standard has separate interfaces and methods for most of these property classes. This makes the version 1.1 standard much more pleasing to use than the version 1.0 standard, where this classification is not really made. But even when working with version 1.0, it is good to be aware of this classification, as it groups properties into classes with very similar qualities.


amsterCHEM
tailor-made engineering software solutions

Slide 34

Special properties:

Special properties must always be supported:

- *pressure* and *temperature*:
  - version 1.0: single value for material object
  - version 1.1: one value for each present phase + Overall
- *fraction* (composition)
- *phaseFraction*
- *flow* (per compound, not relevant in communication with PP)
- *totalFlow* (not relevant in communication with PP)

The special properties we have seen before. They are pressure, temperature, fraction, phaseFraction, flow and totalFlow. All of these properties must be supported by a Material Object in general. When a Material Object however is only used for communication with a Property Package, support of flow and total flow is irrelevant, as a Property Package generally does not set and get their values.

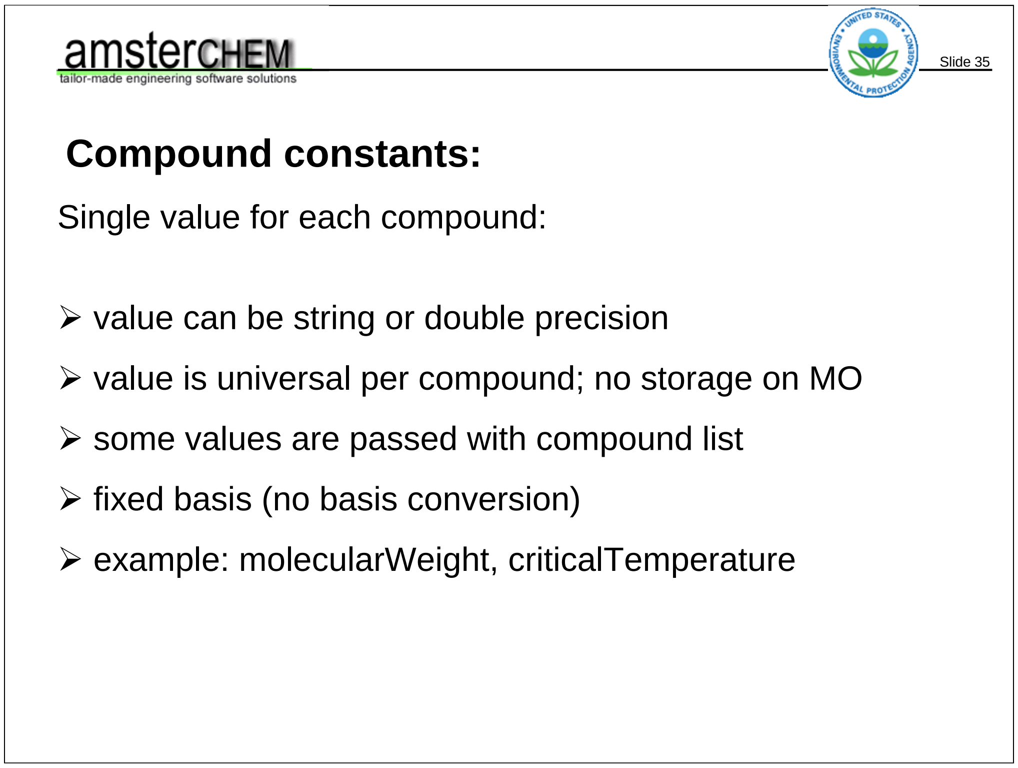
Pressure and Temperature are defined as Material Object Global in version 1.0 thermo, but each phase can have a pressure and temperature in version 1.1 thermo. This situation is not very common, and many Material Objects will behind the screens only implement a single copy of pressure and temperature.

Fraction is the composition for a phase, or for the Overall phase. It is either a mole or mass fraction, depending on the basis.

phaseFraction is the relative amount of a phase that is present. It does not apply to the Overall phase.

Flow is compound flow, with one value for each compound. The sum of flow is totalFlow. TotalFlow is wider supported than flow. Many material objects allow getting flow and totalFlow for a phase, but setting them only for the Overall phase.

Special properties are always inputs to thermodynamic property calculations and are never calculated by a Property Package, with the exception of an equilibrium calculation.


amsterCHEM
tailor-made engineering software solutions

Slide 35

Compound constants:

Single value for each compound:

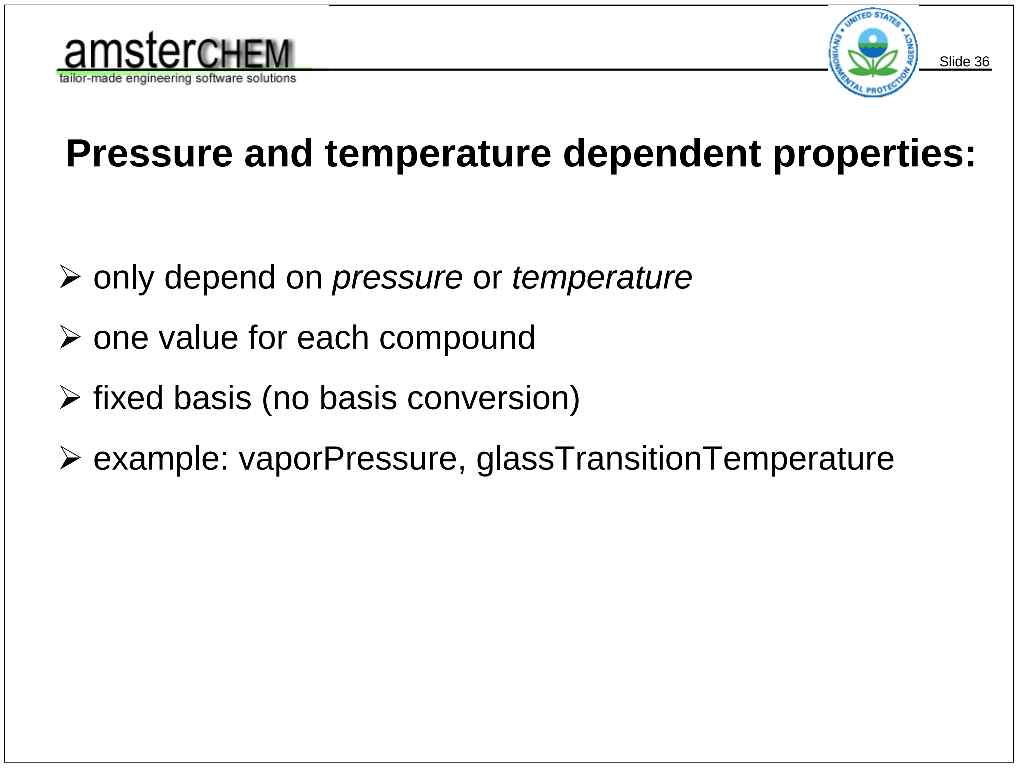
- value can be string or double precision
- value is universal per compound; no storage on MO
- some values are passed with compound list
- fixed basis (no basis conversion)
- example: molecularWeight, criticalTemperature

Compound constants do not depend on pressure, temperature or composition. There are string constants, such as chemicalFormula or CASRegistryNumber, and double precision constants, such as molecularWeight and criticalTemperature.

These values are not stored on the Material Object, there is only one such value per compound.

Some of these values are obtained by the functions that you would call to get the list of supported compounds. These are compound names, CAS numbers, molecular weights and normal boiling points.

Basis conversion does not apply to compound constants. They are always in the same basis. idealGasEnthalpyOfFormationAt25C is always in J/mol, never in J/kg. The standardized compound constants and their unit of measure can be found in the thermo specs.

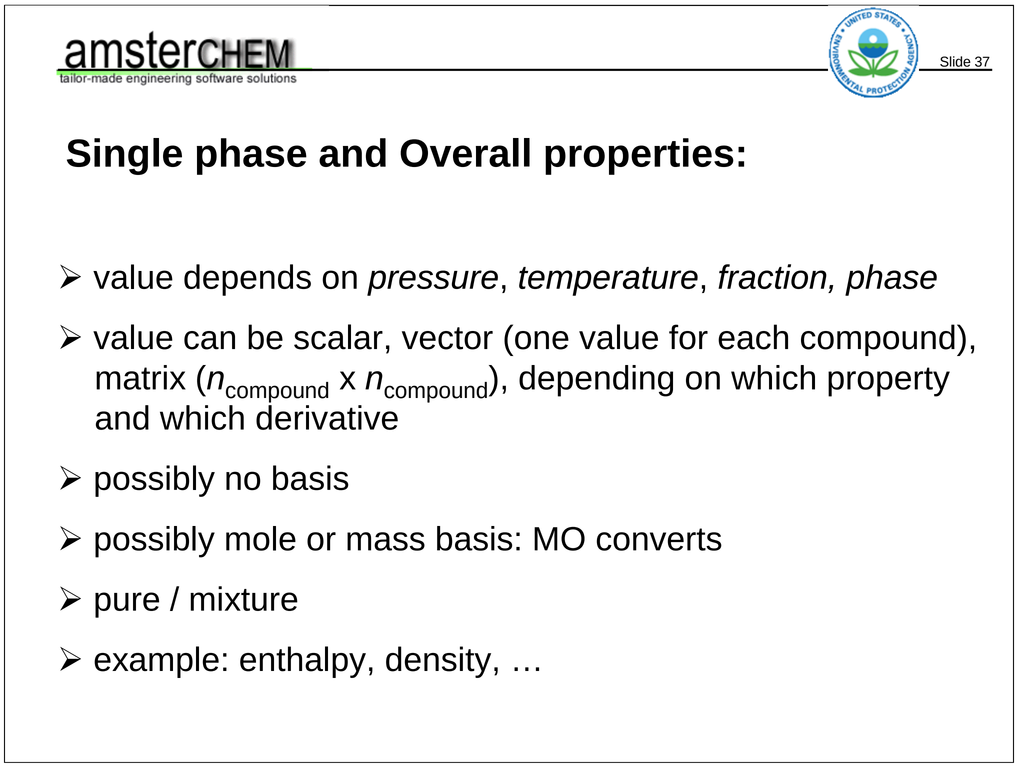

amsterCHEM
tailor-made engineering software solutions

Slide 36

Pressure and temperature dependent properties:

- only depend on *pressure* or *temperature*
- one value for each compound
- fixed basis (no basis conversion)
- example: vaporPressure, glassTransitionTemperature

Pressure and temperature dependent properties do not depend on composition. Pressure dependent properties only depend on pressure. The only valid derivative is that w.r.t. pressure. Temperature dependent properties only depend on temperature. The only valid derivative is that w.r.t. to temperature. These properties always have one value for each compound. Many implementations do not support any pressure dependent properties, but temperature dependent properties are generally available, such as vaporPressure.


amsterCHEM
tailor-made engineering software solutions

Single phase and Overall properties:

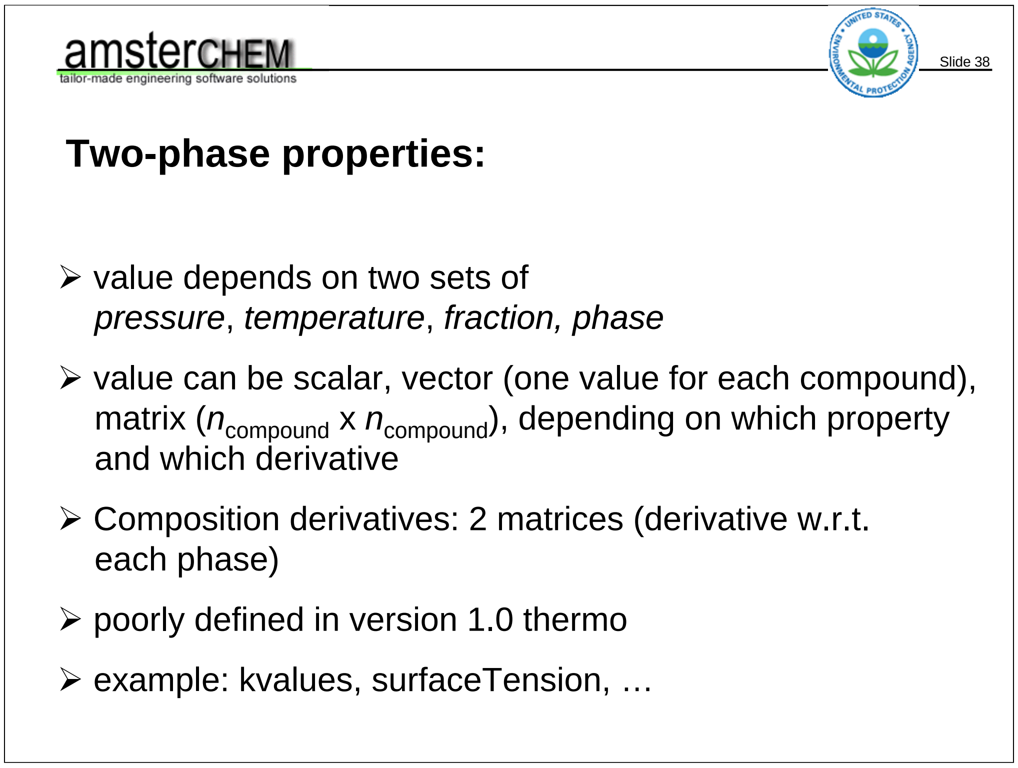
- value depends on *pressure*, *temperature*, *fraction, phase*
- value can be scalar, vector (one value for each compound), matrix ($n_{compound}$ x $n_{compound}$), depending on which property and which derivative
- possibly no basis
- possibly mole or mass basis: MO converts
- pure / mixture
- example: enthalpy, density, …

Single phase and overall properties are the same in version 1.0. The phase identifier for overall properties is Overall. For version 1.1 there are different functions for single phase and overall properties, the functions for overall properties do not take a phase name.

Depending on the property, the value can be a scalar, such as mixture enthalpy, a vector, such as fugacityCoefficient, or a matrix, currently only DiffusionCoefficient. These properties depend on temperature, pressure, composition and the phase specifier. Valid derivatives are those w.r.t. temperature, pressure and composition. Temperature and pressure derivatives do not change the number of values, but composition derivates do. The composition derivative of a scalar is a vector, the composition derivative of a vector is a matrix and the composition derivative of DiffusionCoefficient, - well so far I have not seen support for it.

Basis conversions apply as discussed earlier.

In version 1.0 there is the concept of pure and mixture calculations. For pressure and temperature dependent properties, the proper qualifier would be Pure. For Single phase and overall properties, Mixture is the common qualifier. But for these properties, Pure can be specified as well, which changes the meaning of the property. Pure single-phase properties are not widely supported. In version 1.1 the concept of mixture and pure has disappeared.


amsterCHEM
tailor-made engineering software solutions

Slide 38

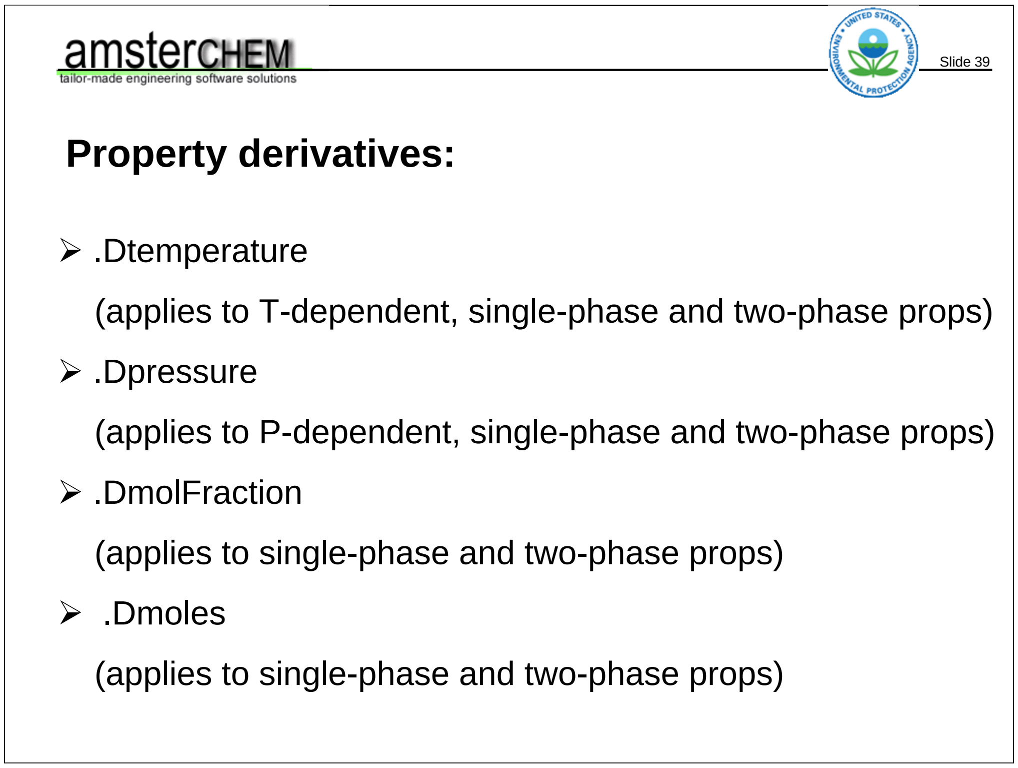
Two-phase properties:

- value depends on two sets of *pressure*, *temperature*, *fraction, phase*
- value can be scalar, vector (one value for each compound), matrix ($n_{compound}$ x $n_{compound}$), depending on which property and which derivative
- Composition derivatives: 2 matrices (derivative w.r.t. each phase)
- poorly defined in version 1.0 thermo
- example: kvalues, surfaceTension, …

Two-phase properties are properties that depend on two phases. Hence, they depend on two temperatures, pressures, compositions and phase identifiers. Two-phase properties are not well defined in version 1.0 thermo. The only two phase properties that are currently defined are surfaceTension and various flavours of K-values. Surface tension is a scalar, Kvalues a vector.

Taking the composition derivatives would make for 2 vectors for surface tension and two matrices for Kvalues. These are the composition derivatives with respect to each phase. This does not hold for temperature and pressure derivatives, that are considered to be w.r.t. a change in pressure or temperature of both phases involved.

In version 1 one would best stay away from two-phase properties as they are poorly defined. For Kvalues, the work-around is simple. K-values are ratios of fugacity coefficients, which are single phase properties. Surface tension in version 1.0 is often implemented as a single-phase liquid property, not taking into account the vapor phase. Liquid-liquid surface tension in version 1.0 cannot be calculated with the original spec. But with the extensions to the spec that allow for phases “Liquid” and “Liquid1” for example, you would calculate the mixture surface tension using a two-phase identifier “LiquidLiquid1”.



Property derivatives:

- .Dtemperature

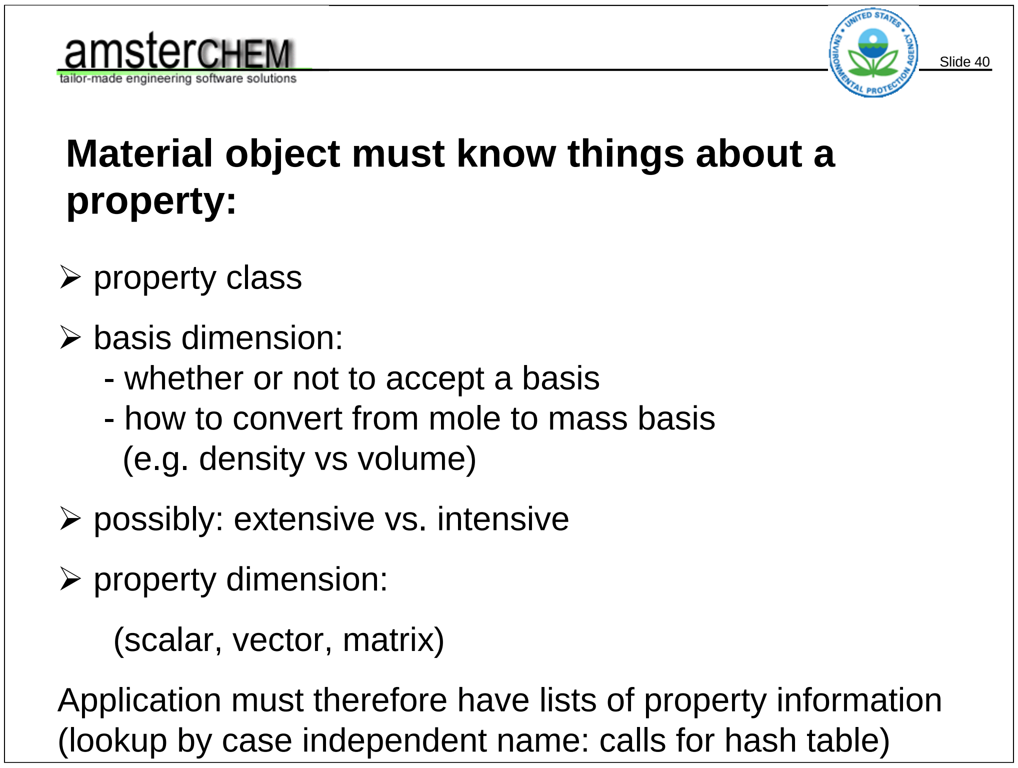
  (applies to T-dependent, single-phase and two-phase props)
- .Dpressure

  (applies to P-dependent, single-phase and two-phase props)
- .DmolFraction

  (applies to single-phase and two-phase props)
- .Dmoles

  (applies to single-phase and two-phase props)

Derivative support in CAPE-OPEN is accomplished by adding a period followed by the derivative name to the property name. Derivatives are defined w.r.t. temperature and pressure and composition. Composition derivatives come in two flavours: mole fraction derivatives and mole number derivatives for 1 mole of substance. Revert to the version 1.1 thermo spec for more details on their difference and use. For the Material Object implementation it is of no consequence; the material object just needs to be able to store them.


amsterCHEM
tailor-made engineering software solutions

Slide 40

Material object must know things about a property:

- property class
- basis dimension:
  - whether or not to accept a basis
  - how to convert from mole to mass basis (e.g. density vs volume)
- possibly: extensive vs. intensive
- property dimension:

  (scalar, vector, matrix)

Application must therefore have lists of property information (lookup by case independent name: calls for hash table)

So far we have learned that the Material Object implementation must know some things about a property. The following items are useful to tabulate and use inside a simulation application.

Not mentioned on this slide is the name of the property. Mind that some properties have a different name in version 1.0 than in version 1.1, so actually if we support both versions, we should store two property names per property.

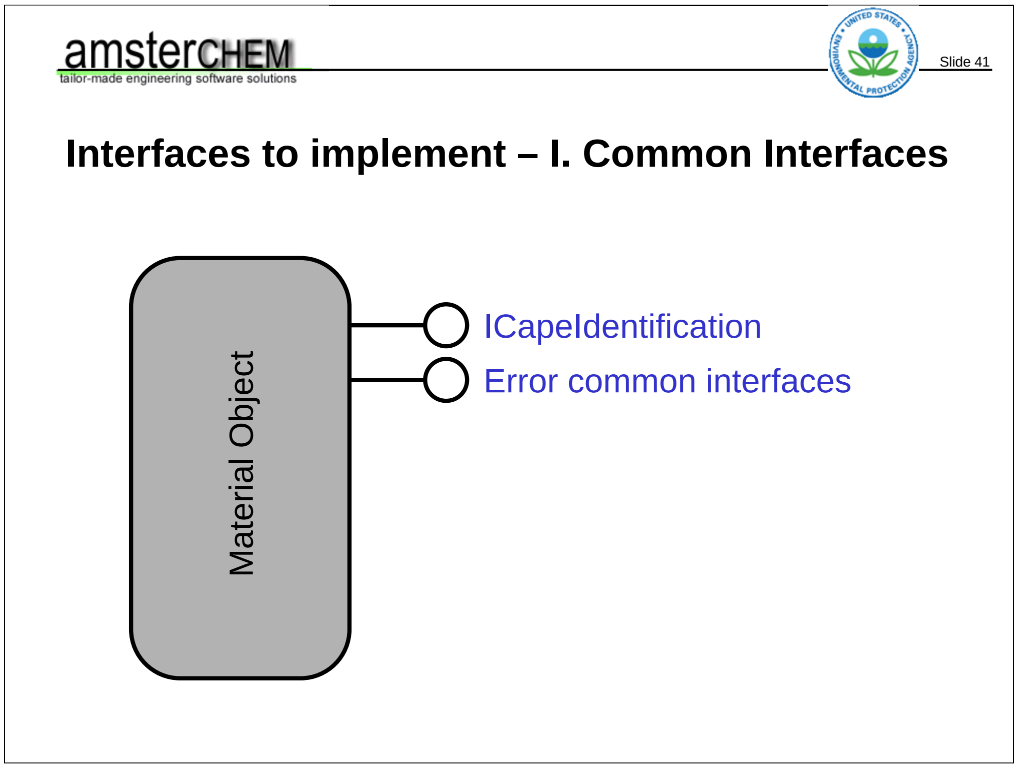
The property class: for version 1.1 implementations this allows compiling lists of supported properties per property class. For version 1.0 implementations the property class will tell you a lot about how to deal with a property.

The conversionOrder as defined before. This tells us whether setting and getting the property requires a basis argument or no basis, and how we go about basis conversions.

One could store whether a property is extensive or intensive. It is useful in some cases; for more information see the definition of composition derivatives in the version 1.1 spec.

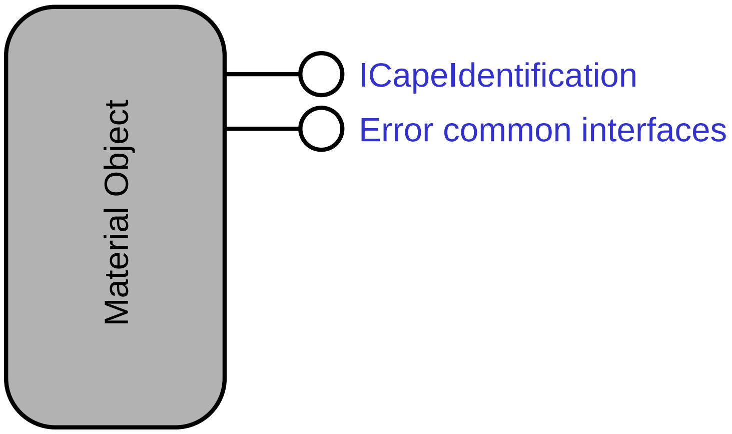
The property dimension: whether it is a scalar, vector, or matrix. Useful for single- and two phase properties, as for the other properties, this is implicitly known by the property class.

These tabulated property definitions must frequently be looked up by the Material Object. A handy implementation is to use a hash table to store the properties by name; this table must use case independent hash keys for the strings though, as in CAPE-OPEN, identifier strings are considered case-insensitive.

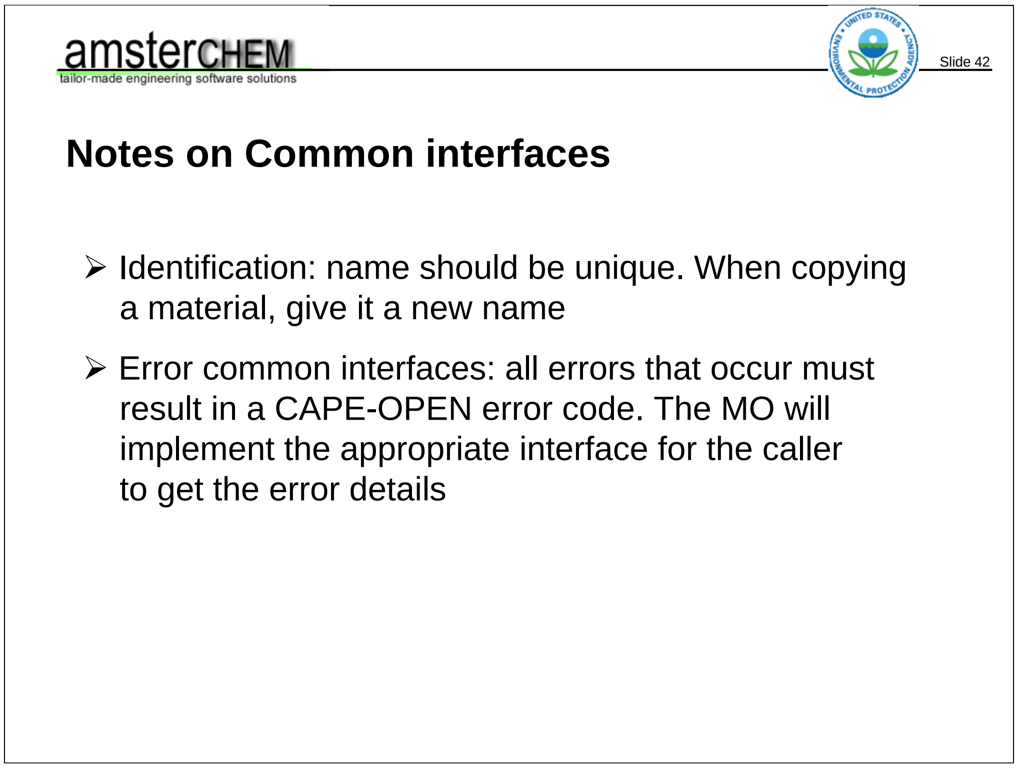


amsterCHEM
tailor-made engineering software solutions
Slide 41

Interfaces to implement – I. Common Interfaces



Now, as a CAPE-OPEN object, the Material Object must implement a number of interfaces. Let start with the interfaces that are implemented by all CAPE-OPEN objects. These are the identification and error interfaces.

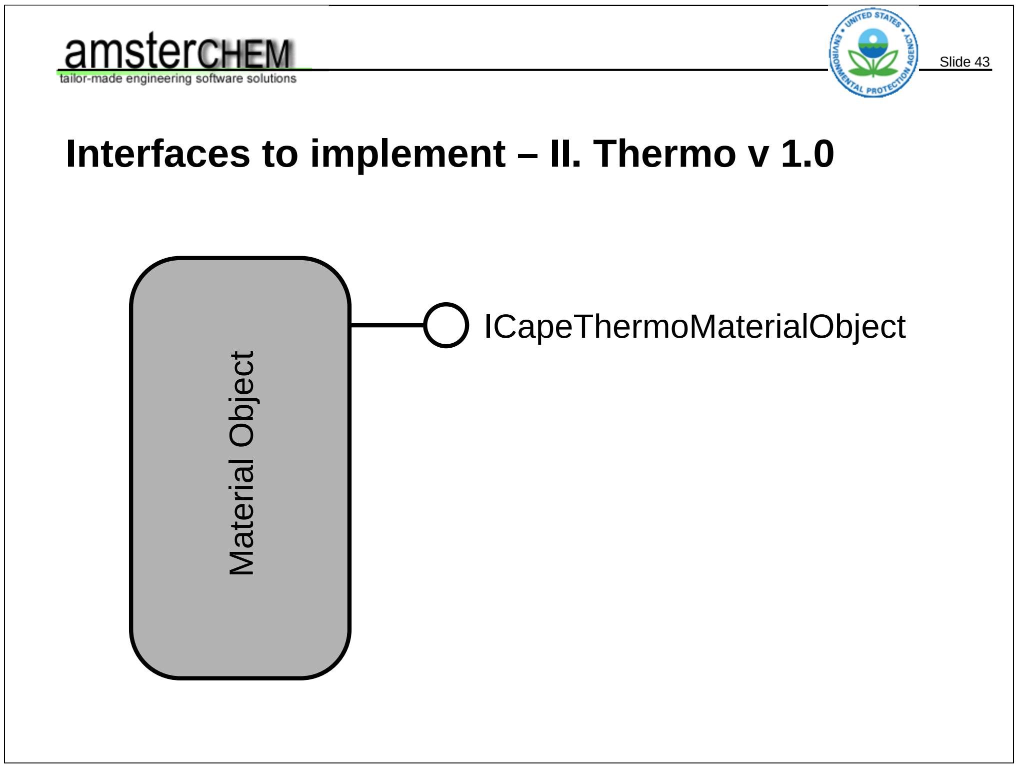

amsterCHEM
tailor-made engineering software solutions

Notes on Common interfaces

- Identification: name should be unique. When copying a material, give it a new name
- Error common interfaces: all errors that occur must result in a CAPE-OPEN error code. The MO will implement the appropriate interface for the caller to get the error details

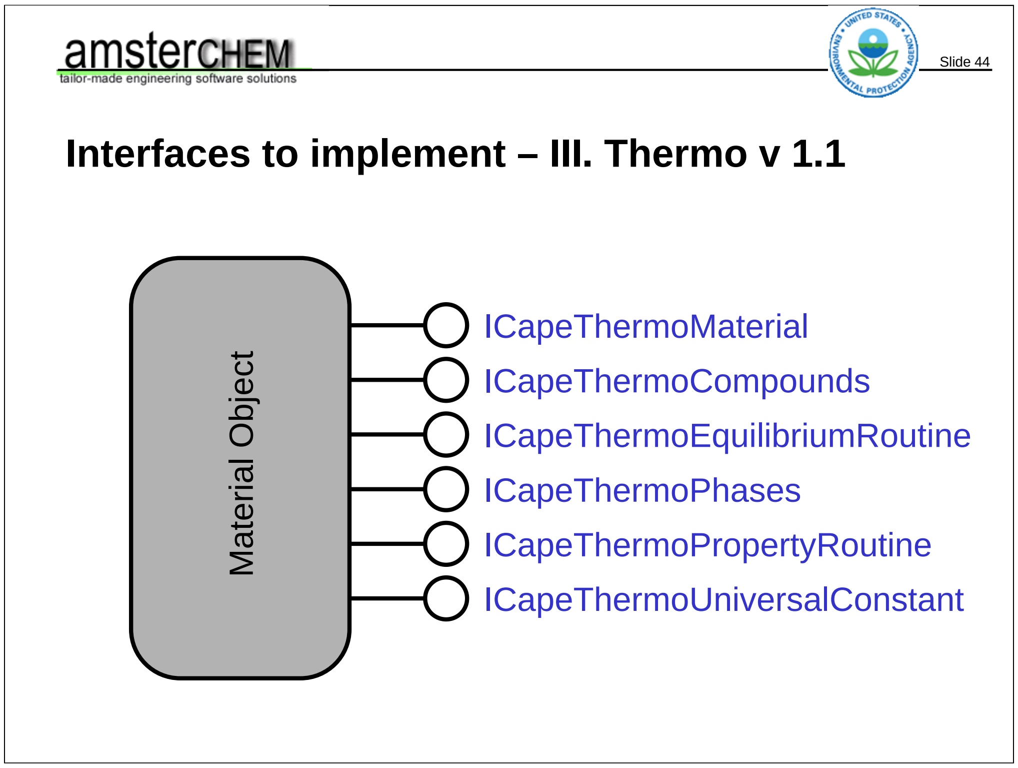
The identification interface exposes a name and a description. The name should be unique throughout the simulation, because the interface suggests it could be used to identify the Material Object.

The error interfaces are used by the caller to retrieve error information when a Material Object returns and error code. At the very least ECapeRoot, ECapeUser and ECapeUnknown must be implemented.



Interfaces to implement – II. Thermo v 1.0

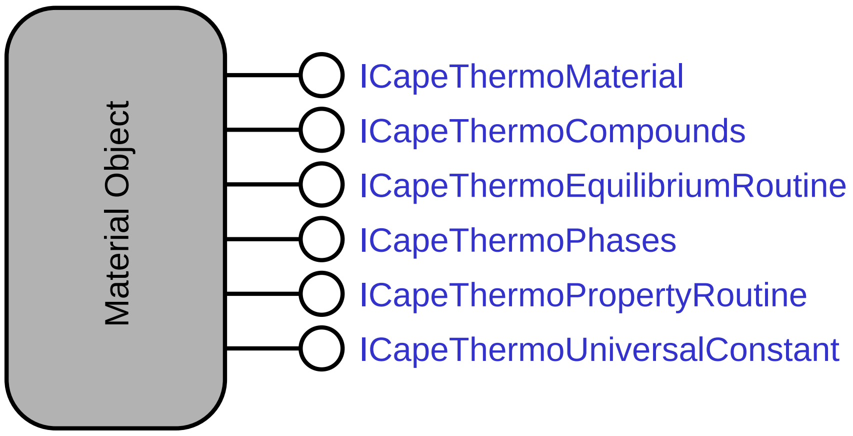
For Material Objects that support version 1.0 thermo, the ICapeThermoMaterialObject interface must be implemented. This interface exposes all functionality of the Material Object.



amsterCHEM
tailor-made engineering software solutions

Slide 44

Interfaces to implement – III. Thermo v 1.1


Material Object
ICapeThermoMaterial
ICapeThermoCompounds
ICapeThermoEquilibriumRoutine
ICapeThermoPhases
ICapeThermoPropertyRoutine
ICapeThermoUniversalConstant

For version 1.1 the functionality is split up in interfaces in a more structural manner. All interfaces shown here, except for ICapeThermoMaterial, are also implemented by a Property Package. The ICapeThermoMaterial interface is specific to a Material Object implementation, and exposes functionality for property storage and present phases.

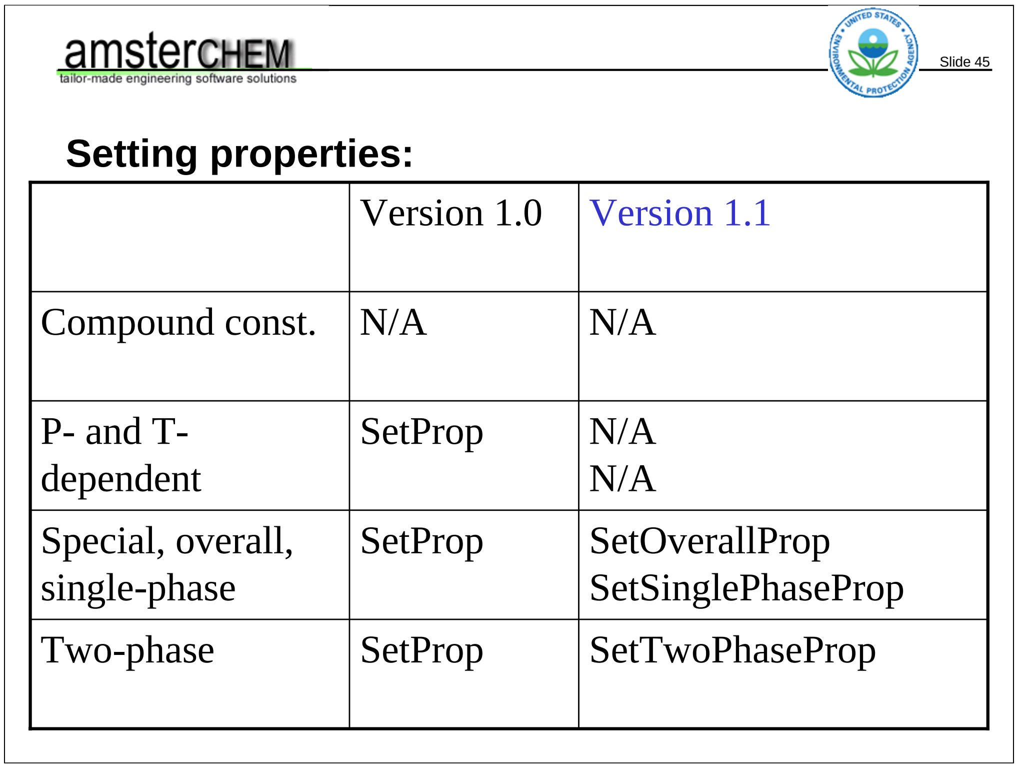
The ICapeThermoCompounds interface exposes the functions for obtaining the supported compounds and their compound constants, temperature dependent properties and pressure dependent properties.

The ICapeThermoEquilibriumRoutine interface allows for performing flash calculations.

The ICapeThermoPhases interace exposes functionality for getting supported phases and their properties. Mind that the present phases are exposed by the ICapeThermoMaterial interface.

The ICapeThermoPropertyRoutine allows for single- and two-phase property calculations.

The ICapeThermoUniversalConstant interface allows to get values like gravitational constant, Avogadro number and gas constant. It is not widely used.


amsterCHEM
tailor-made engineering software solutions

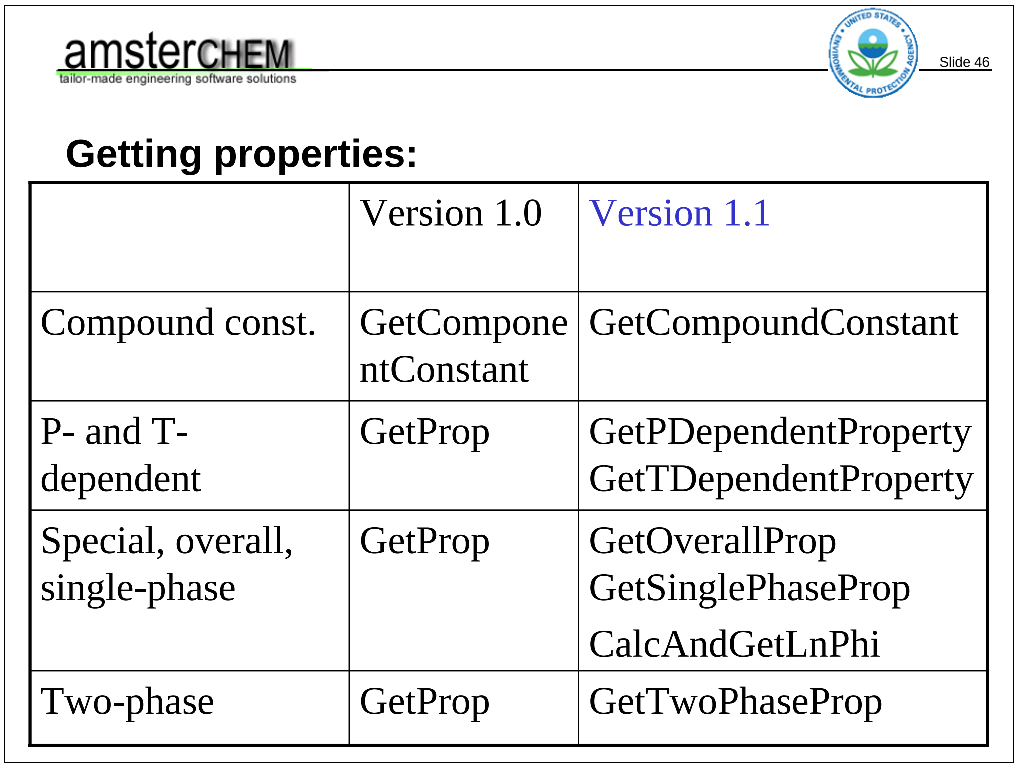
Slide 45

Setting properties:

	Version 1.0	Version 1.1
Compound const.	N/A	N/A
P- and T-dependent	SetProp	N/A N/A
Special, overall, single-phase	SetProp	SetOverallProp SetSinglePhaseProp
Two-phase	SetProp	SetTwoPhaseProp

As you may have noticed, some items in this presentation are blue, and other black. The blue ones we will see supported in the code that we will use for exercises later on.

Here is a quick peek of how properties are set using version 1.0 and version 1.1 thermo. Compound constants are not stored by the material object. Pressure and temperature dependent properties are also not stored, at least in version 1.1 thermo. In version 1.0 thermo they are stored. All storage in version 1.0 thermo uses SetProp. In version 1.1 thermo, the calls are split up with respect to property class.



Getting properties:

	Version 1.0	Version 1.1
Compound const.	GetComponentConstant	GetCompoundConstant
P- and T-dependent	GetProp	GetPDependentProperty GetTDependentProperty
Special, overall, single-phase	GetProp	GetOverallProp GetSinglePhaseProp CalcAndGetLnPhi
Two-phase	GetProp	GetTwoPhaseProp

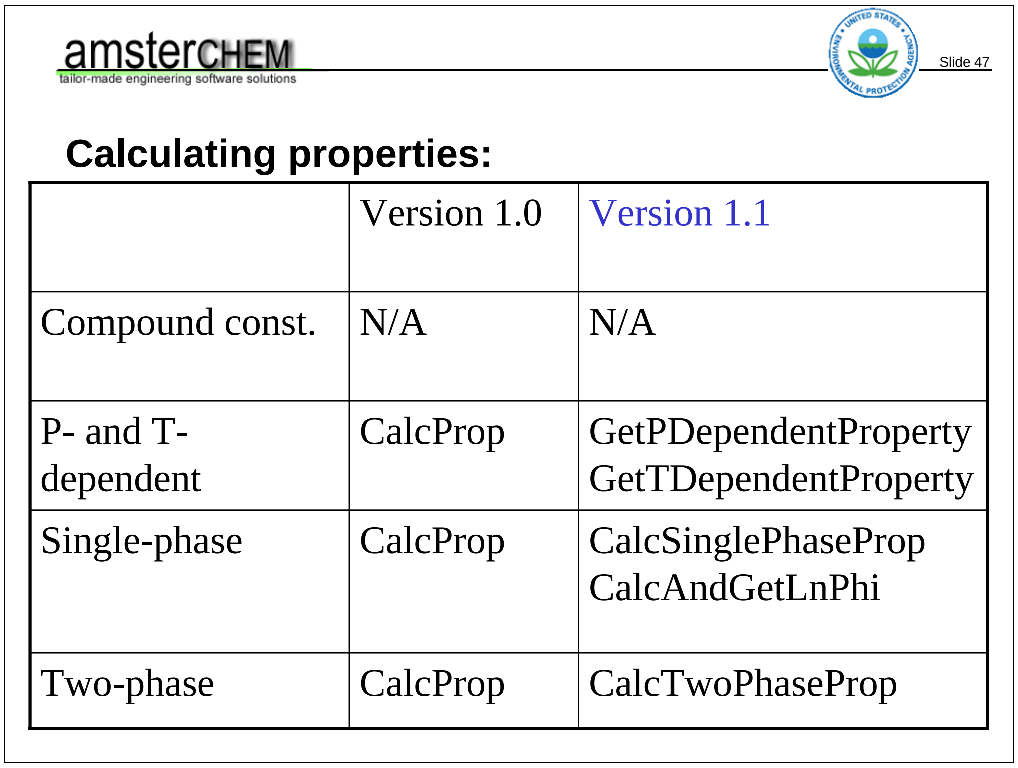
Getting properties is also somewhat different in the two thermo versions. Compound constants are – in both versions – calculated and obtained in the same call.

Pressure and temprature dependent properties in version 1.1 are calculated and obtained in the same call. In version 1.0, you would need to calculate them before you obtain them.

Overall, single phase and two-phase properties always need to be set or calculated before you obtain them. Neither of these calls will calculate the properties for you.

In version 1.0 all properties are obtained with GetProp; in version 1.1 the functionality is split up w.r.t. property classes.

In version 1.1, short-cuts are available to get pressure, temperature and composition in a single call. These are GetTPFraction and GetOverallTPFraction.



amsterCHEM
tailor-made engineering software solutions

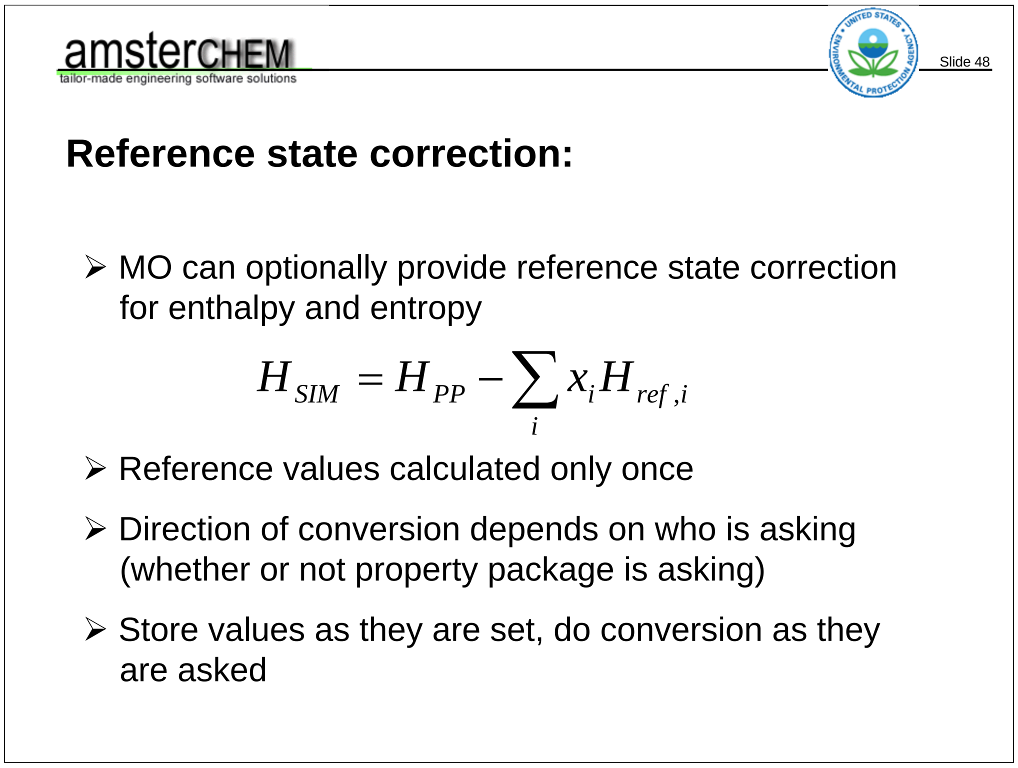
Slide 47

Calculating properties:

	Version 1.0	Version 1.1
Compound const.	N/A	N/A
P- and T-dependent	CalcProp	GetPDependentProperty GetTDependentProperty
Single-phase	CalcProp	CalcSinglePhaseProp CalcAndGetLnPhi
Two-phase	CalcProp	CalcTwoPhaseProp

Compound constants are calculated in the same call as they are obtained. So no special call for calculation. For version 1.1, the following funcitons will both calculate and obtain values: GetPDependentProperty, GetPDependentProperty and CalcAndGetLnPhi. Other functions just calculate properties, after which the calculation results are stored at the Material Object. These are CalcSinglePhaseProp and CalcTwoPhaseProp.

In version 1.0, all property calculations are done using CalcProp.


amsterCHEM
tailor-made engineering software solutions

Slide 48

Reference state correction:

- MO can optionally provide reference state correction for enthalpy and entropy

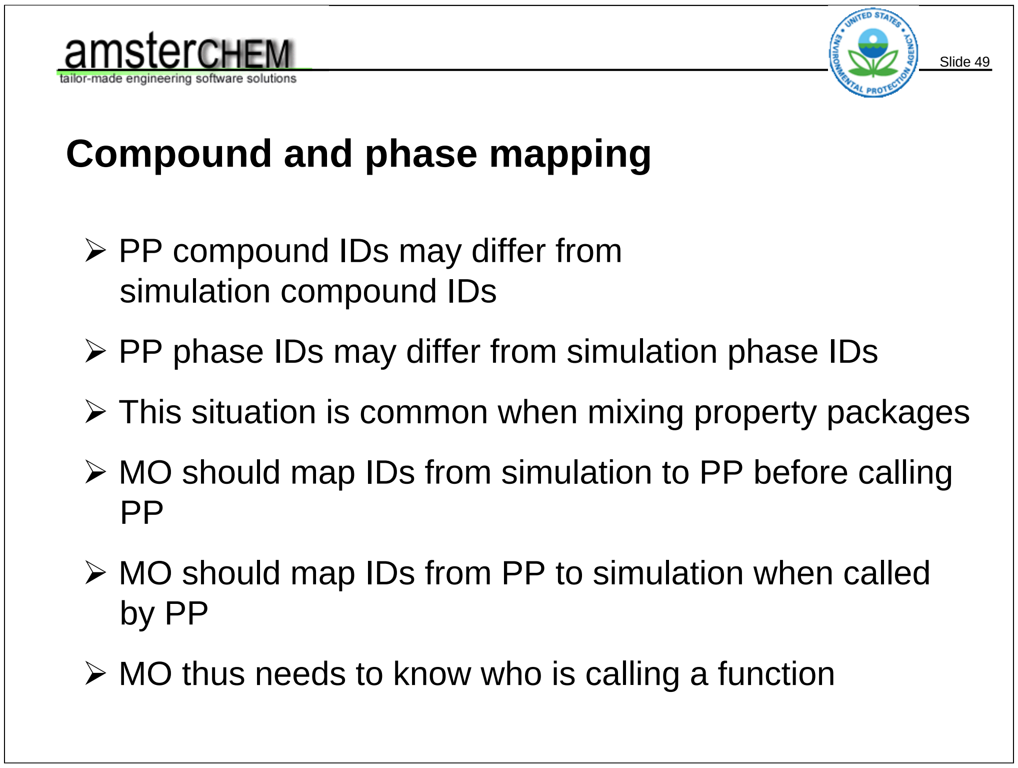
$$H_{SIM} = H_{PP} - \sum_i x_i H_{ref,i}$$

- Reference values calculated only once
- Direction of conversion depends on who is asking (whether or not property package is asking)
- Store values as they are set, do conversion as they are asked

Before the hand-on work, some notes on more exotic features of Material Objects.

It is possible for Material Objects to implement reference state correction for enthalpy and entropy. This involves calculating the pure component enthalpy and entropy values at the reference conditions once. Then – much like the principle of basis conversion – the conversion can be done at the point the properties are obtained. The conversion depends on who is asking. If it is the Property Package that is asking, it should be an inverse reference state correction. So, if the material supports reference state correction, we must store with the properties whether they were set 'before' or 'after' reference state correction.

Note that this implies the Material Object needs to know who is making the call; a SetProp for enthalpy by the Property Package means before reference state correction whereas a SetProp for enthalpy by a Unit Operation means after reference state correction.

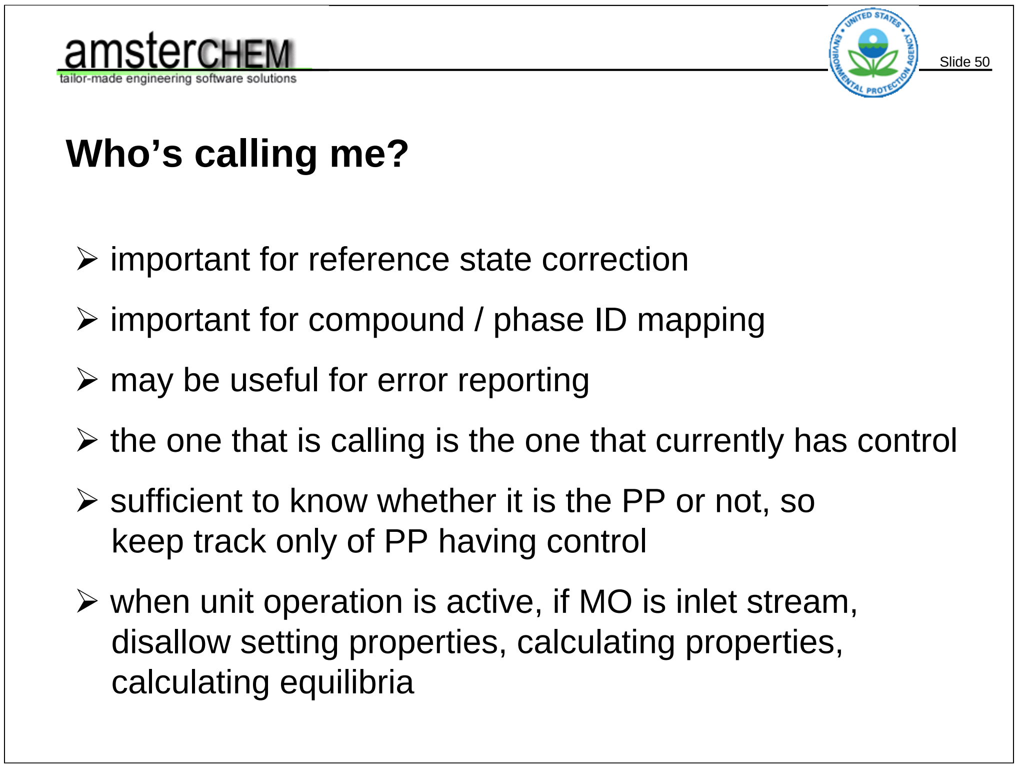


Compound and phase mapping

- PP compound IDs may differ from simulation compound IDs
- PP phase IDs may differ from simulation phase IDs
- This situation is common when mixing property packages
- MO should map IDs from simulation to PP before calling PP
- MO should map IDs from PP to simulation when called by PP
- MO thus needs to know who is calling a function

If a simulation uses multiple underlying thermodynamic systems, it is likely that the phase and compound IDs in one loaded Property Package do not match phase and compound IDs for the same phases and compound in another property package. So in general, the phase and compound property IDs known by the simulation environment and the non-property package PMCs (e.g. the unit operatons) may differ from those exposed by the Property Package. When calling the Property Package, you must use the original phase and compound IDs that the Property Package initially supplied. Hence, some mapping may need to be done. Example, if a compound is called C1 by a property package, and is known in the simulation as METHANE, the Unit Operation may call the material object with a compound list including METHANE. Before the Material Object passes the call on to the Property Package, it must replace METHANE by C1.

Note that this requires that the Property Package needs to know who is calling the function; it needs to know what set of names to map from and to.


amsterCHEM
tailor-made engineering software solutions

Slide 50

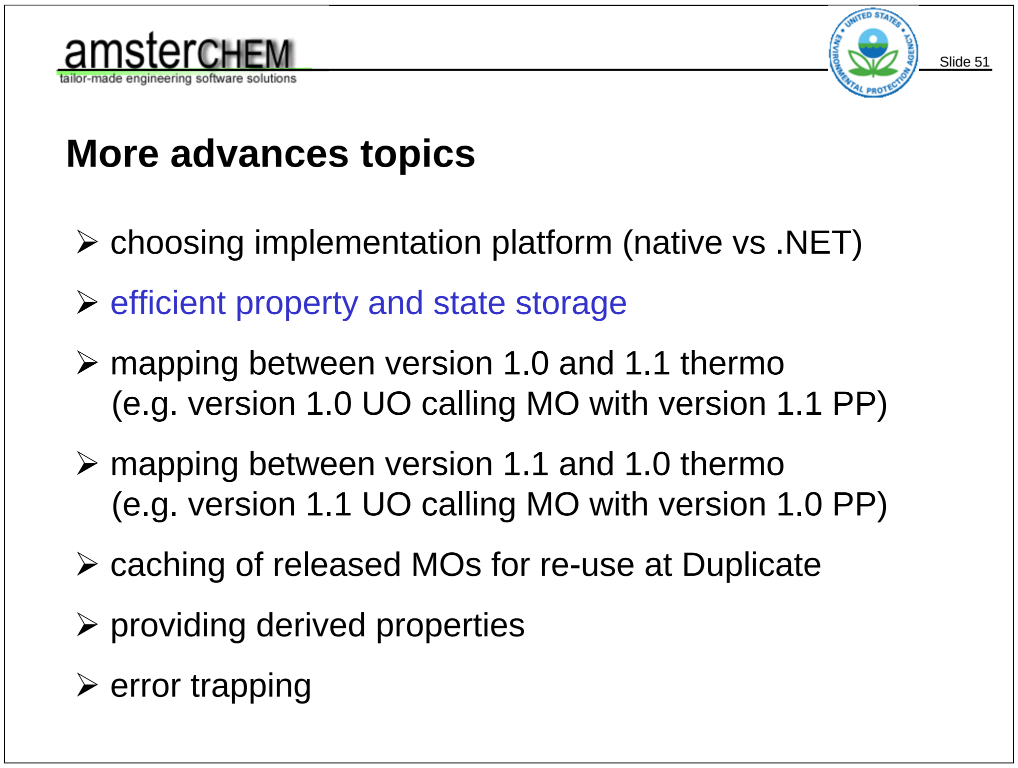
Who’s calling me?

- important for reference state correction
- important for compound / phase ID mapping
- may be useful for error reporting
- the one that is calling is the one that currently has control
- sufficient to know whether it is the PP or not, so keep track only of PP having control
- when unit operation is active, if MO is inlet stream, disallow setting properties, calculating properties, calculating equilibria

The last two slides indicated that the Material Object may need to know who is calling a function; it is important for proper implementation of reference state correction as well as compound and phase ID mapping. It may also be useful for error reporting.

The caller of the Material Object is the software component that currently has execution control. Often it is enough to know whether it is the Property Package calling the Material Object, or not. So we may need to keep track on whether the Property Package currently has execution control. This can be accomplished by making a note of this before calling any function in the Property Package. This note can be discarded as soon as the Property Package function call returns.

Sometimes it is useful to know that it is the Unit Operation making the call. Specifically for Material Objects representing inlet streams. One could then disallow setting or calculating properties or flashes, as a Unit Operation is not supposed to do that on inlet streams.


amsterCHEM
tailor-made engineering software solutions

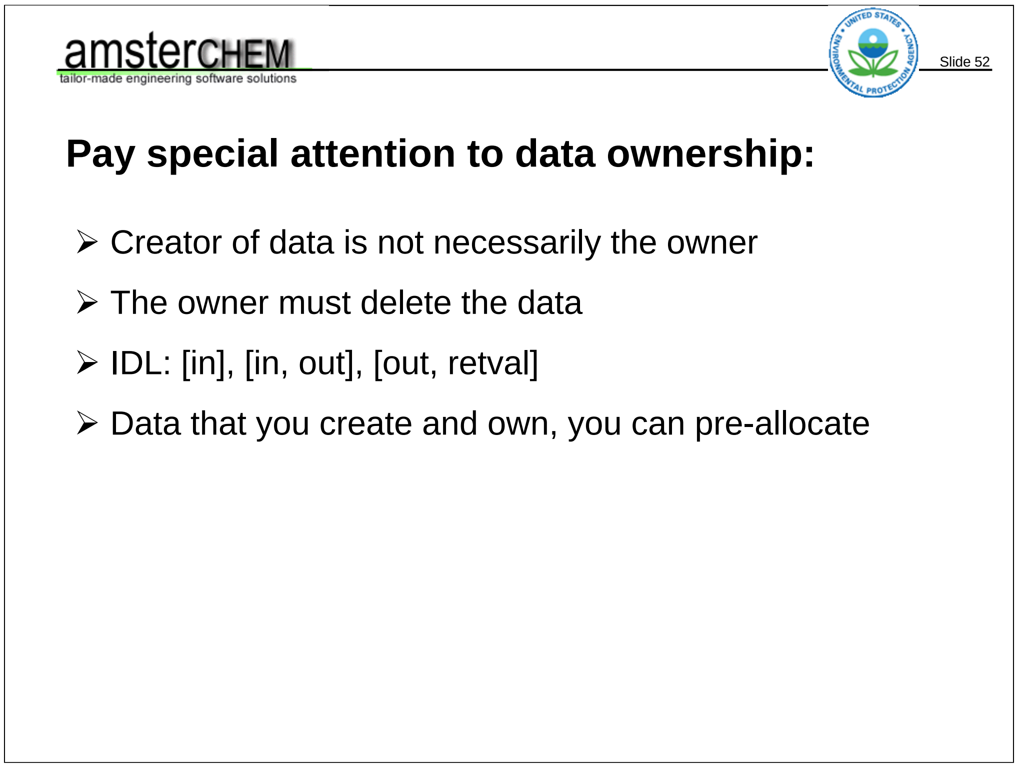
Slide 51

More advances topics

- choosing implementation platform (native vs .NET)
- efficient property and state storage
- mapping between version 1.0 and 1.1 thermo
  (e.g. version 1.0 UO calling MO with version 1.1 PP)
- mapping between version 1.1 and 1.0 thermo
  (e.g. version 1.1 UO calling MO with version 1.0 PP)
- caching of released MOs for re-use at Duplicate
- providing derived properties
- error trapping

This slide presents a list of further topics of interest when implementing Material Objects for production platforms. With the test application that is implemented for this course we do not have to be so picky, as it supports only one version of thermo, and its efficiency is not critical.

…..


amsterCHEM
tailor-made engineering software solutions

Slide 52

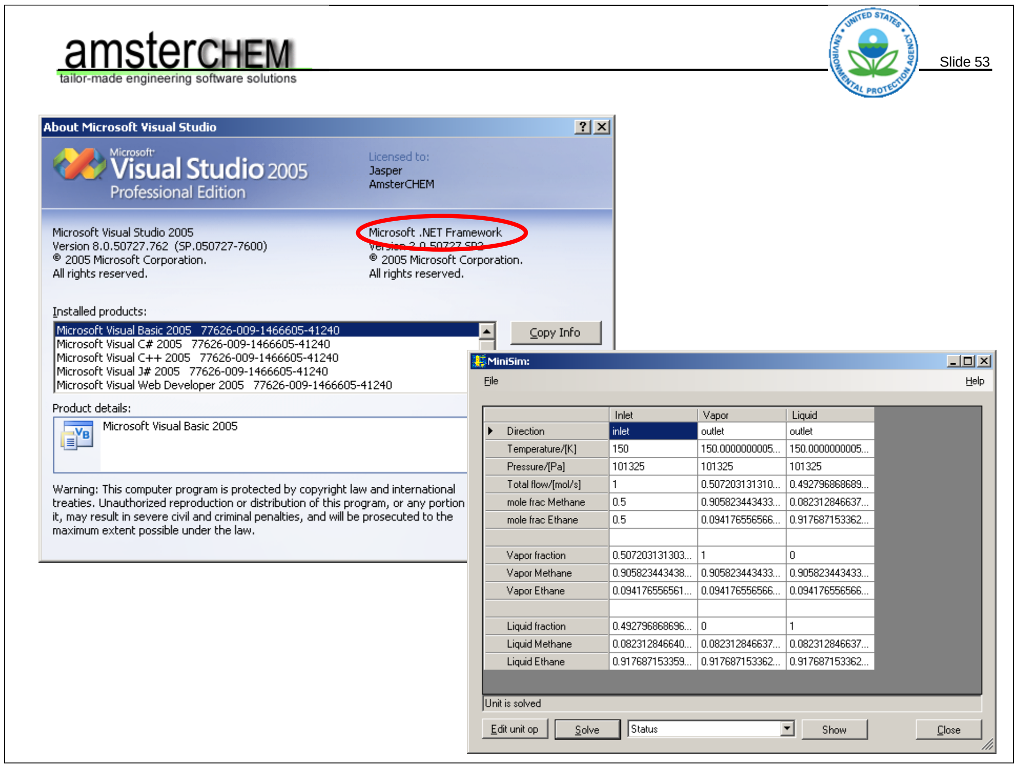
Pay special attention to data ownership:

- Creator of data is not necessarily the owner
- The owner must delete the data
- IDL: [in], [in, out], [out, retval]
- Data that you create and own, you can pre-allocate

As a final remark: as we will be using .NET for our implementation, we will not be bothered with looking at data ownership. Most commonly, and implementation will not be .NET based (or VB 6 based) so we are responsible for keeping track of data ownership. Data ownership is important to keep track of, as it defines who must destroy the data. Fail to destroy the data and you have a memory leak. Destroy the data multiple times and you will find your application crashes.

CAPE-OPEN (and COM) define the interaction between different software components. The software component that owns data must destroy it. This is not necessarily the component that created the data. For example, if a PMC asks a material for a property value, the material creates it, but the PMC owns it.

If you have a closer look at the IDL, you will find that each argument is marked with [in], [in, out] or [out, retval]. This defines the ownership. [in] and [out,retval] data is owned by the caller. For [in, out] data, the data that comes in is owned by the implementer, and the data that goes out is owned by the caller. Hence, for a function like GetCompoundList, we find that all arguments are marked [in, out]. As we typically allocate the output data in the routine implementation, this means we must free whatever is in the arguments when the routine gets invoked.


amsterCHEM
tailor-made engineering software solutions
Slide 53
About Microsoft Visual Studio
Microsoft Visual Studio 2005
Professional Edition
Copy Info
MiniSim:
File
Help
Edit unit op
Solve
Status
Show
Close

Licensed to:
Jasper
AmsterCHEM

Microsoft Visual Studio 2005
Version 8.0.50727.762 (SP.050727-7600)
® 2005 Microsoft Corporation.
All rights reserved.

Microsoft .NET Framework
Version 2.0.50727 SP2
® 2005 Microsoft Corporation.
All rights reserved.

Installed products:

Microsoft Visual Basic 2005 77626-009-1466605-41240
Microsoft Visual C# 2005 77626-009-1466605-41240
Microsoft Visual C++ 2005 77626-009-1466605-41240
Microsoft Visual J# 2005 77626-009-1466605-41240
Microsoft Visual Web Developer 2005 77626-009-1466605-41240

Product details:

Microsoft Visual Basic 2005

Warning: This computer program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program, or any portion it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

	Inlet	Vapor	Liquid
Direction	inlet	outlet	outlet
Temperature/[K]	150	150.0000000005...	150.0000000005...
Pressure/[Pa]	101325	101325	101325
Total flow/[mol/s]	1	0.507203131310...	0.492796868689...
mole frac Methane	0.5	0.905823443433...	0.082312846637...
mole frac Ethane	0.5	0.094176556566...	0.917687153362...
Vapor fraction	0.507203131303...	1	0
Vapor Methane	0.905823443438...	0.905823443433...	0.905823443433...
Vapor Ethane	0.094176556561...	0.094176556566...	0.094176556566...
Liquid fraction	0.492796868696...	0	1
Liquid Methane	0.082312846640...	0.082312846637...	0.082312846637...
Liquid Ethane	0.917687153359...	0.917687153362...	0.917687153362...

Unit is solved

Sample code used for exercises is MiniSim (.NET based code); available for CO-LaN members.