Create a Heater

In [1]:
```
import pyomo.environ as pe
from pyomo.common.config import ConfigBlock, ConfigValue, In
from idaes.core import (ControlVolume0DBlock,
                        declare_process_block_class,
                        EnergyBalanceType,
                        MomentumBalanceType,
                        MaterialBalanceType,
                        UnitModelBlockData,
                        useDefault,
                        FlowsheetBlock)
from idaes.core.util.config import is_physical_parameter_block
from methanol_param_VLE import PhysicalParameterBlock
from idaes.core.util.misc import add_object_reference
```

In [2]:
```
def make_control_volume(unit, name, config):
    if config.dynamic is not False:
        raise ValueError('IdealGasIsentropcCompressor does not support dynamics')
    if config.has_holdup is not False:
        raise ValueError('IdealGasIsentropcCompressor does not support holdup')

    control_volume = ControlVolume0DBlock(default={"property_package": config.property_package,
                                                   "property_package_args": config.property_package

    setattr(unit, name, control_volume)

    control_volume.add_state_blocks(has_phase_equilibrium=config.has_phase_equilibrium)
    control_volume.add_material_balances(balance_type=config.material_balance_type,
                                         has_phase_equilibrium=config.has_phase_equilibrium)
    control_volume.add_total_enthalpy_balances(has_heat_of_reaction=False,
                                               has_heat_transfer=True,
                                               has_work_transfer=False)
    control_volume.add_total_pressure_balances(has_pressure_change=False)
```

In [3]:
```
def make_config_block(config):
    config.declare("material_balance_type",
        ConfigValue(default=MaterialBalanceType.componentPhase, domain=In(MaterialBalanceType)))
    config.declare("energy_balance_type",
        ConfigValue(default=EnergyBalanceType.enthalpyTotal, domain=In([EnergyBalanceType.enthalpyT
    config.declare("momentum_balance_type",
        ConfigValue(default=MomentumBalanceType.pressureTotal, domain=In([MomentumBalanceType.press
    config.declare("has_phase_equilibrium",
        ConfigValue(default=False, domain=In([False])))
    config.declare("has_pressure_change",
        ConfigValue(default=False, domain=In([False])))
    config.declare("property_package",
        ConfigValue(default=useDefault, domain=is_physical_parameter_block))
    config.declare("property_package_args",
        ConfigBlock(implicit=True))
```

In [4]:
```
@declare_process_block_class("Heater")
class HeaterData(UnitModelBlockData):
    CONFIG = UnitModelBlockData.CONFIG()
    make_config_block(CONFIG)

    def build(self):
        super(HeaterData, self).build()

        make_control_volume(self, "control_volume", self.config)

        self.add_inlet_port()
        self.add_outlet_port()

        add_object_reference(self, 'heat', self.control_volume.heat[0.0])
```

In [5]:
```
m = pe.ConcreteModel()
m.fs = fs = FlowsheetBlock(default={"dynamic": False})
fs.properties = props = PhysicalParameterBlock(default={'Cp': 0.038056, 'valid_phase': 'Vap'})

fs.heater = Heater(default={"property_package": props, 'has_phase_equilibrium': False})
fs.heater.inlet.flow_mol.fix(1)
fs.heater.inlet.mole_frac[0, 'CH3OH'].fix(0.25)
fs.heater.inlet.mole_frac[0, 'CH4'].fix(0.25)
fs.heater.inlet.mole_frac[0, 'H2'].fix(0.25)
fs.heater.inlet.mole_frac[0, 'CO'].fix(0.25)
fs.heater.inlet.pressure.fix(0.1)
fs.heater.inlet.temperature.fix(3)
fs.heater.heat.fix(5)

opt = pe.SolverFactory('ipopt')
opt.options['linear_solver'] = 'mumps'
res = opt.solve(m, tee=False)
print(res.solver.termination_condition)
fs.heater.outlet.display()
```

```
optimal
outlet : Size=1
    Key  : Name        : Value
    None :    flow_mol : {0.0: 1.0}
         :   mole_frac : {(0.0, 'CH3OH'): 0.25, (0.0, 'CH4'): 0.25, (0.0, 'CO'): 0.25, (0.0, 'H2'
): 0.25}
         :    pressure : {0.0: 0.1}
         : temperature : {0.0: 4.313853268866933}
```

In [ ]: