## Excessive Infiltration

### Description

def description

return "Excessive infiltration around the building envelope occurs by the unintentional introduction of outside air into a building, typically through cracks in the building envelope and through use of windows and doors. Infiltration is driven by pressure differences between indoors and outdoors of the building caused by wind and by air buoyancy forces known commonly as the stack effect. Excessive infiltration can affect thermal comfort, indoor air quality, heating and cooling demand, and moisture damage of building envelope components. This fault is categorized as a fault that occur in the building envelope during the operation stage. This fault measure is based on a physical model where certain parameter(s) is changed in EnergyPlus to mimic the faulted operation; thus simulates an excessive infiltration by modifying ZoneInfiltration:DesignFlowRate or ZoneInfiltration:EffectiveLeakageArea objects in EnergyPlus. The fault intensity (F) is defined as the ratio of excessive infiltration around the building envelope compared to the non-faulted condition."

end

### Modeler Description

def modeler\_description

return "The user input of the ratio of excessive infiltration is applied to one of either four variables (Design Flow Rate, Flow per Zone Floor Area, Flow per Exterior Surface Area, Air Changes per Hour) in ZoneInfiltration:DesignFlowRate object and one variable (Effective Air Leakage Area) in ZoneInfiltration:EffectiveLeakageArea depending on the user’s choice of infiltration implementation method to impose fault over the original (non-faulted) configuration. The modified value (Infil\_m) is calculated as Infil\_m = Infil\_o \* (1+F), where Infil\_o is the original value defined in the infiltration object and F is the percentage of excessive infiltration. The time required for the fault to reach the full level is only required when the user wants to model fault evolution. If the fault evolution is not necessary for the user, it can be defined as zero and F will be imposed as a step function with the user defined value. However, by defining the time required for the fault to reach the full level, fault starting month/date/time and fault ending month/date/time, the adjustment factor AF is calculated at each time step starting from the starting month/date/time to gradually impose F based on the user specified time frame. AF is calculated as follows, AF\_current = AF\_previous + dt/tau where AF\_current is the adjustment factor calculated based on the previously calculated adjustment factor (AF\_previous), simulation timestep (dt) and the time required for the fault to reach the full level (tau)."

end

### Measure Type

OpenStudio Measure

**Taxonomy**

Envelope.Infiltration

### Arguments

def arguments(model)

args = OpenStudio::Ruleset::OSArgumentVector.new

#make a choice argument for model objects

thermalzone\_handles = OpenStudio::StringVector.new

thermalzone\_display\_names = OpenStudio::StringVector.new

#putting model object and names into hash

thermalzone\_args = model.getSpaces

thermalzone\_args\_hash = {}

thermalzone\_args.each do |thermalzone\_arg|

thermalzone\_args\_hash[thermalzone\_arg.name.to\_s] = thermalzone\_arg

end

#looping through sorted hash of model objects

thermalzone\_args\_hash.sort.map do |key,value|

thermalzone\_handles << value.handle.to\_s

thermalzone\_display\_names << key

end

thermalzone\_display\_names << "\*Entire Building\*"

#add building to string vector with space type

building = model.getBuilding

thermalzone\_handles << building.handle.to\_s

#make a choice argument for space type

thermalzone = OpenStudio::Ruleset::OSArgument::makeChoiceArgument("thermalzone", thermalzone\_handles, thermalzone\_display\_names)

thermalzone.setDisplayName("Apply the Measure to a Specific Space Type or to the Entire Model.")

thermalzone.setDefaultValue("\*Entire Building\*") #if no space type is chosen this will run on the entire building

args << thermalzone

#make an argument for excessive infiltration percentage

space\_infiltration\_increase\_percent = OpenStudio::Ruleset::OSArgument::makeDoubleArgument("space\_infiltration\_increase\_percent",true)

space\_infiltration\_increase\_percent.setDisplayName("Space Infiltration Increase (%).")

space\_infiltration\_increase\_percent.setDefaultValue(20.0)

args << space\_infiltration\_increase\_percent

##################################################

#Parameters for transient fault modeling

#make a double argument for the time required for fault to reach full level

time\_constant = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('time\_constant', false)

time\_constant.setDisplayName('Enter the time required for fault to reach full level [hr]')

time\_constant.setDefaultValue(0) #default is zero

args << time\_constant

#make a double argument for the start month

start\_month = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('start\_month', false)

start\_month.setDisplayName('Enter the month (1-12) when the fault starts to occur')

start\_month.setDefaultValue(6) #default is June

args << start\_month

#make a double argument for the start date

start\_date = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('start\_date', false)

start\_date.setDisplayName('Enter the date (1-28/30/31) when the fault starts to occur')

start\_date.setDefaultValue(1) #default is 1st day of the month

args << start\_date

#make a double argument for the start time

start\_time = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('start\_time', false)

start\_time.setDisplayName('Enter the time of day (0-24) when the fault starts to occur')

start\_time.setDefaultValue(9) #default is 9am

args << start\_time

#make a double argument for the end month

end\_month = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('end\_month', false)

end\_month.setDisplayName('Enter the month (1-12) when the fault ends')

end\_month.setDefaultValue(12) #default is Decebmer

args << end\_month

#make a double argument for the end date

end\_date = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('end\_date', false)

end\_date.setDisplayName('Enter the date (1-28/30/31) when the fault ends')

end\_date.setDefaultValue(31) #default is last day of the month

args << end\_date

#make a double argument for the end time

end\_time = OpenStudio::Ruleset::OSArgument::makeDoubleArgument('end\_time', false)

end\_time.setDisplayName('Enter the time of day (0-24) when the fault ends')

end\_time.setDefaultValue(23) #default is 11pm

args << end\_time

##################################################

return args

end

### Initial Condition

#Checking the number of infiltration object defined in the model,

runner.registerInitialCondition("The initial model contained #{space\_infiltration\_objects.size + space\_infiltration\_ela\_objects.size} space infiltration objects.")

#If there is no infiltration object defined in the model,

runner.registerInitialCondition("The initial model did not contain any space infiltration objects.")

### Final Condition

#Report number of infiltration object modified,

runner.registerFinalCondition("#{altered\_instances} space infiltration objects in the model were altered.")

### Not Applicable

#When no infiltration objects were modified,

runner.registerAsNotApplicable("No space infiltration objects were found in the specified space type(s) and no life cycle costs were requested.")

### Warning

#When fault intensity (increase of infiltration) is defined too small,

runner.registerWarning("A space infiltration increase percentage of #{space\_infiltration\_increase\_percent} percent is abnormally low.")

#When fault intensity (increase of infiltration) is defined too high,

runner.registerWarning("A space infiltration increase percentage of #{space\_infiltration\_increase\_percent} percent is abnormally high.")

#When infiltration object’s field is not properly defined,

runner.registerWarning("'#{instance.name}' is used by one or more instances and has no load values.")

### Error

#When space type was not chosen,

runner.registerError("No space type was chosen.")

#When chosen space type is not found in the model,

runner.registerError("The selected space type with handle '#{handle}' was not found in the model. It may have been removed by another measure.")

### Information

* Works with,
  + ZoneInfiltration:DesignFlowRate
  + ZoneInfiltration:EffectiveLeakageArea.
* Future refinement item is,
  + "Space" input option (with drop down menu) instead of "SpaceType" option. LCC cost codes are not used.

### Code Outline

* Define arguments (zone where fault occurs, percentage of increased infiltration).
  + Check whether fault intensity (increase of infiltration) is reasonably defined within 0-100.
  + Modify infiltration objects based on “space” object.
    - Read and replace infiltration method defined in ZoneInfiltration:DesignFlowRate.
      * Append EMS code that calculates the adjustment factor (AF)
      * Replace infiltration values based on user defined fault intensity (F)
        + designFlowRate schedule = (schedule value) \* (1 + F/100\*AF)
        + flowperSpaceFloorArea schedule = (schedule value) \* (1 + F/100\*AF)
        + flowperExteriorSurfaceArea schedule = (schedule value) \* (1 + F/100\*AF)
        + flowperExteriorWallArea schedule = (schedule value) \* (1 + F/100\*AF)
        + airChangesperHour schedule = (schedule value) \* (1 + F/100\*AF)
    - Read and replace infiltration method defined in ZoneInfiltration:EffectiveLeakageArea.
      * Append EMS code that calculates the adjustment factor (AF)
      * Replace infiltration values based on user defined fault intensity
        + effectiveAirLeakageArea schedule= (schedule value) \* (1 + F/100\*AF)

### Tests

* Test invalid user argument values to make sure measure fails gracefully
* Test different infiltration methods.