

External Interface(s) Application Guide

Guide for using EnergyPlus with External
Interface(s)

(specific to the Building Controls Virtual Test Bed
a.k.a BCVTB)

Date: October 11, 2011

COPYRIGHT © 1996-2011 The Board of Trustees of the University of Illinois and the Regents of the University of California through the Ernest Orlando Lawrence Berkeley National Laboratory pending approval of the US Department of Energy.

All Rights Reserved. No part of this material may be reproduced or transmitted in any form or by any means without the prior written permission of the University of Illinois or the Ernest Orlando Lawrence Berkeley National Laboratory.

EnergyPlus is a Trademark of the US Department of Energy.

TABLE OF CONTENTS

External Interface.....	1
Introduction	1
Algorithm for data exchange.....	1
References.....	2
Examples	3
Architecture of System	3
Figure 1: Architecture of the BCVTB with the EnergyPlus client (black) and other clients (grey).	3
Table 1. Overview of the EnergyPlus objects used in Examples	4
XML Syntax.....	4
Example 1: Interface using ExternalInterface:Schedule.....	5
Creating the EnergyPlus idf file.....	5
Creating the configuration file	6
Creating the Ptolemy model	7
Figure 2: System model in the BCVTB.....	8
Figure 3: Configuration of the Simulator actor that calls EnergyPlus on Windows	8
Figure 4: Configuration of the Simulator actor that calls EnergyPlus on Mac OS X and on Linux.	9
Example 2: Interface using ExternalInterface:Actuator	9
Creating the EnergyPlus idf file.....	9
Creating the configuration file	10
Example 3: Interface using ExternalInterface:Variable	11
Creating the EnergyPlus idf file.....	11
Creating the configuration file	12

External Interface

Introduction

The ExternalInterface in EnergyPlus allows coupling EnergyPlus with the Building Controls Virtual Test Bed (BCVTB). The BCVTB is a software environment, developed by Lawrence Berkeley National Laboratory and available from <http://simulationresearch.lbl.gov/bcvtb>. It allows expert users to couple different simulation programs for distributed simulation or for a real-time simulation that is connected to a building control system. For example, the BCVTB allows simulation of the building envelope and HVAC system in EnergyPlus and the control logic in MATLAB/Simulink, while exchanging data between the two programs as they simulate. EnergyPlus users who want to use the BCVTB will need to download the BCVTB from Lawrence Berkeley National Laboratory.

Algorithm for data exchange

The process in which at least two simulators solve initial-value differential equations that are coupled to each other is called co-simulation. Various algorithms are possible for the data exchange. In the BCVTB, data are exchanged between its client programs, including EnergyPlus, using a fixed synchronization time step. There is no iteration between the clients. In the co-simulation literature, this coupling scheme is referred to as *quasi-dynamic coupling*, *loose coupling* or *ping-pong coupling* [Hensen 1999, Zhai and Chen 2005].

The algorithm for exchanging data is as follows: Suppose we have a system with two clients, with client 1 being EnergyPlus and client 2 being, for example, the Simulink program from Mathworks. Suppose each client solves an initial-value ordinary differential equation that is coupled to the differential equation of the other client. Let $N \in \mathbb{N}$ denote the number of time steps and let $k \in \{1, K, N\}$ denote the time steps. We will use the subscripts 1 and 2 to denote the state variable and the function that computes the next state variable of the simulator 1 and 2, respectively.

The simulator 1 computes, for $k \in \{1, K, N-1\}$, the sequence

$$x_1(k+1) = f_1(x_1(k), x_2(k))$$

and, similarly, the simulator 2 computes the sequence

$$x_2(k+1) = f_2(x_2(k), x_1(k))$$

with initial conditions $x_1(0) = x_{1,0}$ and $x_2(0) = x_{2,0}$.

To advance from time k to $k+1$, each simulator uses its own time integration algorithm. At the end of the time step, the simulator 1 sends the new state $x_1(k+1)$ to the BCVTB and it receives the state $x_2(k+1)$ from the BCVTB. The same procedure is done with the simulator 2. The BCVTB synchronizes the data in such a way that it does not matter which of the two simulators is called first.

In comparison to numerical methods of differential equations, this scheme is identical to an explicit Euler integration, which is an integration algorithm that computes for an ordinary differential equation with specified initial values,

$$dx/dt = h(x),$$

$$x(0) = x_0,$$

on the time interval $t \in [0, 1]$, the following sequence:

- Step 0:** Initialize counter $k=0$ and number of steps $N \in \mathbb{N}$.
Set initial state $x(k) = x_0$ and set time step $\Delta t = 1/N$.
- Step 1:** Compute new state $x(k+1) = x(k) + h(x(k)) \Delta t$.
Replace k by $k+1$.
- Step 2:** If $k=N$ stop, else go to Step 1.

In the situation where the differential equation is solved using co-simulation, the above algorithm becomes

- Step 0:** Initialize counter $k=0$ and number of steps $N \in \mathbb{N}$.
Set initial state $x_1(k) = x_{1,0}$ and $x_2(k) = x_{2,0}$. Set the time step $\Delta t = 1/N$.
- Step 1:** Compute new states
 $x_1(k+1) = x_1(k) + f_1(x_1(k), x_2(k)) \Delta t$, and
 $x_2(k+1) = x_2(k) + f_2(x_2(k), x_1(k)) \Delta t$.
Replace k by $k+1$.
- Step 2:** If $k=N$ stop, else go to Step 1.

This algorithm is implemented in the BCVTB. Note that there is no iteration between the two simulators.

References

- Hensen, Jan L. M. 1999. "A comparison of coupled and de-coupled solutions for temperature and air flow in a building." *ASHRAE Transactions* 105 (2): 962–969.
- Zhai, Zhiqiang John, and Qingyan Yan Chen. 2005. "Performance of coupled building energy and CFD simulations." *Energy and Buildings* 37 (4): 333–344.

Examples

Architecture of System

The figure below shows the architecture of the connection between EnergyPlus and the BCVTB. The black objects are explained in this application guide, whereas the grey items are not specific to EnergyPlus and are explained in the BCVTB documentation. The BCVTB connects to the external interface in EnergyPlus. In the external interface, the input/output signals that are exchanged between the BCVTB and EnergyPlus are mapped to EnergyPlus objects. The subject of this External Interface Application Guide is how to configure this mapping and how to use these objects. For a detailed explanation of the grey items, we refer to the BCVTB documentation.

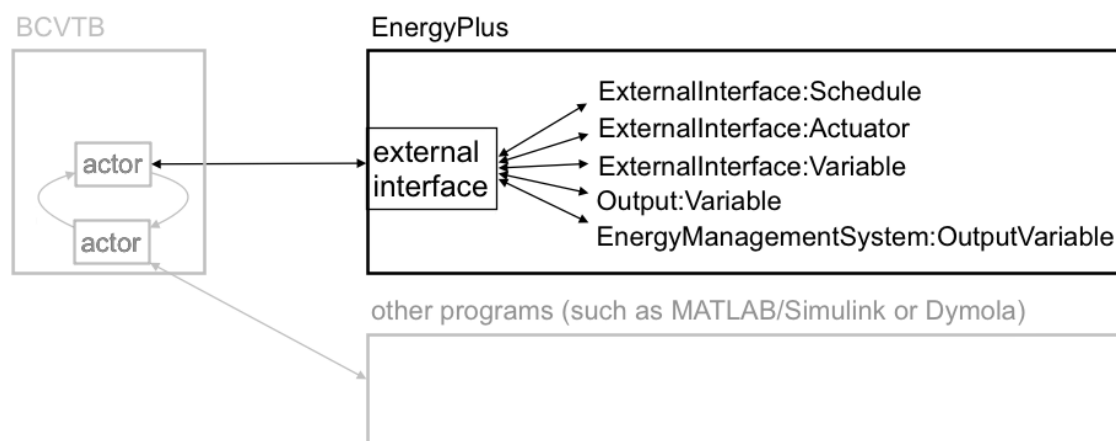


Figure 1: Architecture of the BCVTB with the EnergyPlus client (black) and other clients (grey).

The external interface can map to three EnergyPlus input objects called ExternalInterface:Schedule, ExternalInterface:Actuator and ExternalInterface:Variable. The ExternalInterface:Schedule can be used to overwrite schedules, and the other two objects can be used in place of Energy Management System (EMS) actuators and EMS variables. The objects have similar functionality as the objects Schedule:Compact, EnergyManagementSystem:Actuator and EnergyManagementSystem:GlobalVariable, except that their numerical value is obtained from the external interface at the beginning of each zone time step, and will remain constant during this zone time step.

Compared to EnergyManagementSystem:Actuator, the object ExternalInterface:Actuator has an optional field called "initial value." If a value is specified for this field, then this value will be used during the warm-up period and the system sizing. If unspecified, then the numerical value for this object will only be used during the time stepping. Since actuators always overwrite other objects (such as a schedule), all these objects have values that are defined during the warm-up and the system sizing even if no initial value is specified. For the objects ExternalInterface:Schedule and ExternalInterface:Variable, the field "initial value" is required, and its value will be used during the warm-up period and the system-sizing.

ExternalInterface:Variable is a global variable from the point of view of the EMS language. Thus, it can be used within any EnergyManagementSystem:Program in the same way as an EnergyManagementSystem:GlobalVariable or an EnergyManagementSystem:Sensor can be used.

Although variables of type ExternalInterface:Variable can be assigned to EnergyManagementSystem:Actuator objects, for convenience, there is also an object called

ExternalInterface:Actuator. This object behaves identically to EnergyManagementSystem:Actuator, with the following exceptions:

- Its value is assigned by the external interface.
- Its value is fixed during the zone time step because this is the synchronization time step for the external interface.

The external interface can also map to the EnergyPlus objects Output:Variable and EnergyManagementSystem:OutputVariable. These objects can be used to send data from EnergyPlus to the BCVTB at each zone time step.

We will now present examples that use all of these objects. The following table shows which EnergyPlus features are used in the examples, which are all distributed with the BCVTB installation that can be obtained from the LBNL web site. Note – these examples are NOT distributed with EnergyPlus installation because you need the special software to make them work.

Table 1. Overview of the EnergyPlus objects used in Examples

	Example 1	Example 2	Example 3
ExternalInterface:Schedule	x		
ExternalInterface:Actuator		X	
ExternalInterface:Variable			x
Output:Variable	x	X	x
EnergyManagementSystem:OutputVariable			x

To configure the data exchange, the following three steps are required from the user:

- 1) Create an EnergyPlus idf file.
- 2) Create an xml file that defines the mapping between EnergyPlus and BCVTB variables.
- 3) Create a Ptolemy model.

These steps are described in the examples below. Prior to discussing the examples, we will explain the syntax of the xml configuration file that defines how data are mapped between the external interface and EnergyPlus

XML Syntax

This section describes the syntax of the xml file that configures the data mapping between EnergyPlus and the external interface.

The data mapping between EnergyPlus and the external interface is defined in an xml file called variables.cfg. This file needs to be in the same directory as the EnergyPlus idf file.

The file has the following header:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BCVTB-variables SYSTEM "variables.dtd">
```

Following the header is an element of the form

```
<BCVTB-variables>
</BCVTB-variables>
```

This element will contain child elements that define the variable mapping. In between the element tags, a user needs to specify how the exchanged data is mapped to EnergyPlus objects. Hence, the order of these elements matter, and it need to be the same as the order of the elements in the input and output signal vector of the BCVTB actor that calls EnergyPlus. The exchanged variables are declared in elements that are called “variable” and have an attribute “source.” As described above, the external interface can send data to ExternalInterface:Schedule, ExternalInterface:Actuator, ExternalInterface:Variable. For these

objects, the “source” attribute needs to be set to “Ptolemy,” because they are computed in Ptolemy. The xml elements for these objects look as follows:

For ExternalInterface:Schedule, use

```
<variable source="Ptolemy">
  <EnergyPlus schedule="NAME"/>
</variable>
```

where NAME needs to be the EnergyPlus schedule name. For ExternalInterface:Actuator, use

```
<variable source="Ptolemy">
  <EnergyPlus actuator="NAME" />
</variable>
```

where NAME needs to be the EnergyPlus actuator name. For ExternalInterface:Variable, use

```
<variable source="Ptolemy">
  <EnergyPlus variable="NAME"/>
</variable>
```

where NAME needs to be the EnergyPlus Energy Runtime Language (Erl) variable name.

The external interface can also read data from any Output:Variable and EnergyManagementSystem:OutputVariable. For these objects, set the “source” attribute to “EnergyPlus,” because they are computed by EnergyPlus. The read an Output:Variable, use

```
<variable source="EnergyPlus">
  <EnergyPlus name="NAME" type="TYPE"/>
</variable>
```

where NAME needs to be the EnergyPlus “Variable Name” (such as ZONE/SYS AIR TEMP) and TYPE needs to be the EnergyPlus “Key Value” (such as ZONE ONE). To read an EnergyManagementSystem:OutputVariable, use

```
<variable source="EnergyPlus">
  <EnergyPlus name="EMS" type="TYPE"/>
</variable>
```

i.e., the attribute “name” must be EMS, and the attribute “type” must be set to the EMS variable name.

Complete examples of these xml files are presented below.

Example 1: Interface using ExternalInterface:Schedule

In this example, a controller that is implemented in the BCVTB computes the room temperature set points for cooling and heating. The example can be found in the BCVTB distribution in the folder examples/ePlusX-schedule, where X stands for the EnergyPlus version number.

Suppose we need to send from the BCVTB to EnergyPlus a schedule value, and from EnergyPlus to the BCVTB an output variable at each zone time step. This can be accomplished by using an object of type ExternalInterface:Schedule and an object of type Output:Variable.

To interface EnergyPlus using the EMS feature, the following three items are needed:

- An object that instructs EnergyPlus to activate the external interface.
- EnergyPlus objects that write data from the external interface to the EMS.
- A configuration file to configure the data exchange.

Creating the EnergyPlus idf file

The EnergyPlus idf file contains the following objects to activate and use the external interface:

- An object that instructs EnergyPlus to activate the external interface.

- An object of type ExternalInterface:Schedule. The external interface will write its values to these objects at each zone time-step.
- Objects of type Output:Variable. Any EnergyPlus output variable can be read by the external interface.

The code below shows how to declare these objects.

To activate the external interface, we use:

```
ExternalInterface,          !- Object to activate the external interface
PtolemyServer;            !- Name of external interface
```

To enter schedules to which the external interface writes, we use:

```
! Cooling schedule. This schedule is set directly by the external interface.
! During warm-up and system-sizing, it is fixed at 24 degC.
ExternalInterface:Schedule,
  TSetCoo,                !- Name
  Temperature,            !- ScheduleType
  24;                     !- Initial value, used during warm-up

! Heating schedule. This schedule is set directly by the external interface.
! During warm-up and system-sizing, it is fixed at 20 degC.
ExternalInterface:Schedule,
  TSetHea,                !- Name
  Temperature,            !- ScheduleType
  20;                     !- Initial value, used during warm-up
```

These schedules can be used as other EnergyPlus schedules. In this example, they are used to change a thermostat setpoint:

```
ThermostatSetpoint: DualSetpoint,
  DualSetPoint,          !- Name
  BCVTB-SP-TH,          !- Heating Setpoint Temperature Schedule Name
  BCVTB-SP-TC;          !- Cooling Setpoint Temperature Schedule Name
```

We also want to read from EnergyPlus output variables, which we declare as

```
Output:Variable,
  TSetHea,              !- Key Value
  Schedule Value,      !- Variable Name
  TimeStep;            !- Reporting Frequency

Output:Variable,
  TSetCoo,              !- Key Value
  Schedule Value,      !- Variable Name
  TimeStep;            !- Reporting Frequency
```

To specify that data should be exchanged every 15 minutes of simulation time, enter in the idf file the section

```
Timestep,
  4;                    !- Number of Timesteps per Hour
```

Creating the configuration file

Note that we have not yet specified the order of the elements in the signal vector that is exchanged between EnergyPlus and the BCVTB. This information is specified in the file variables.cfg. The file variables.cfg needs to be in the same directory as the EnergyPlus idf file. For the objects used in the section above, the file looks like

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BCVTB-variables SYSTEM "variables.dtd">
<BCVTB-variables>
  <!-- The next two elements send the set points to E+ -->
  <variable source="Ptolemy">
    <EnergyPlus schedule="TSetHea"/>
  </variable>
  <variable source="Ptolemy">
    <EnergyPlus schedule="TSetCoo"/>
  </variable>
  <!-- The next two elements receive the outdoor and zone air temperature from E+ -->
  <variable source="EnergyPlus">
    <EnergyPlus name="ENVIRONMENT" type="OUTDOOR DRY BULB"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="ZSF1" type="ZONE/SYS AIR TEMPERATURE"/>
  </variable>
  <!-- The next two elements receive the schedule value as an output from E+ -->
  <variable source="EnergyPlus">
    <EnergyPlus name="TSetHea" type="Schedule Value"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="TSetCoo" type="Schedule Value"/>
  </variable>
</BCVTB-variables>

```

This file specifies that the actor in the BCVTB that calls EnergyPlus has an input vector with two elements that are computed by Ptolemy (Ptolemy is the name of the software on which the BCVTB is based) and sent to EnergyPlus, and that it has an output vector with four elements that are computed by EnergyPlus and sent to Ptolemy. The order of the elements in each vector is determined by the order in the above XML file. Hence, the input vector that contains the signals sent to EnergyPlus has elements

```

TSetHea
TSetCoo

```

and the output vector that contains values computed by EnergyPlus has elements

```

Environment (Outdoor drybulb temperature)
ZSF1 (ZONE/SYS AIR TEMPERATURE)
TSetHea (Schedule Value)
TSetCoo (Schedule Value)

```

Creating the Ptolemy model

To start EnergyPlus from the BCVTB, you will need to create a Ptolemy model.

The model `bcbtb/example/ePlus40-schedule/system-windows.xml` that is part of the BCVTB installation and that is shown below may be used as a starting point. (For Mac and Linux, use the file `system.xml`.) In this example, the time step is 15 minutes and the simulation period is four days.

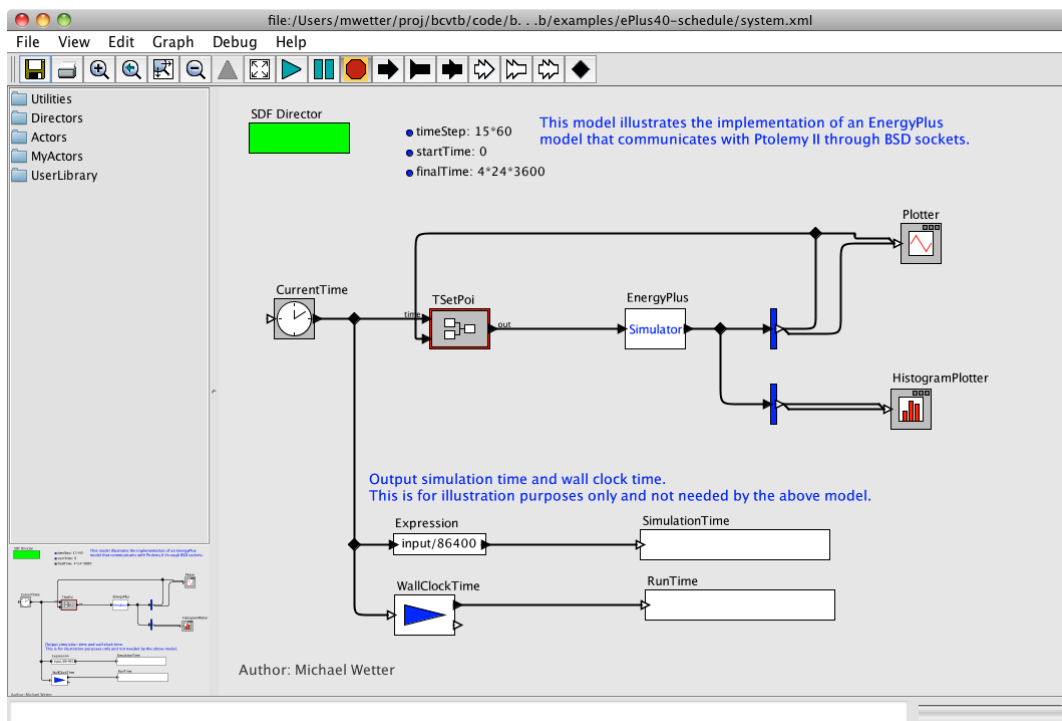


Figure 2: System model in the BCVTB.

In this model, the Simulator actor that calls EnergyPlus is configured for Windows as follows:

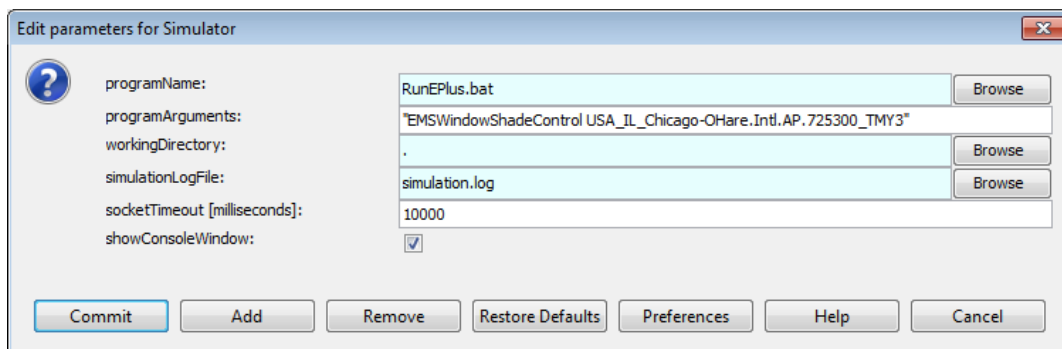


Figure 3: Configuration of the Simulator actor that calls EnergyPlus on Windows.

Hence, it calls the file "RunEPlus.bat," with arguments "EMSWindowShadeControl USA_IL_Chicago-OHare.Intl.AP.725300_TMY3." The working directory is the current directory and the console output is written to the file simulation.log. If EnergyPlus does not communicate with the BCVTB within 10 seconds, the BCVTB will terminate the connection. (See <http://simulationresearch.lbl.gov/bcvtb> for more detailed documentation about how to configure a BCVTB model that communicates with other programs.)

For Mac OS X and Linux, the configuration is similar:

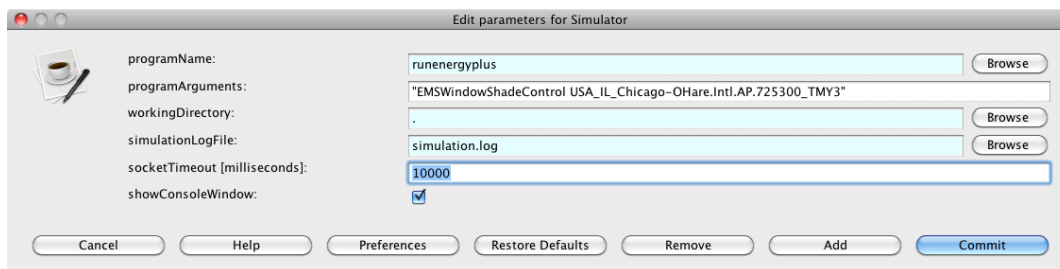


Figure 4: Configuration of the Simulator actor that calls EnergyPlus on Mac OS X and on Linux.

This completes the configuration.

Example 2: Interface using ExternalInterface:Actuator

In this example, a shading controller with a finite state machine is implemented in the BCVTB. Inputs to the controller are the outside temperature and the solar radiation that is incident on the window. The output of the controller is the shading actuation signal.

This example describes how to set up EnergyPlus to exchange data between the BCVTB and EnergyPlus, using an Energy Management System (EMS) actuator. The example can be found in the BCVTB distribution in the folder `examples/ePlusX-actuator`, where `X` stands for the EnergyPlus version number.

The object of type `ExternalInterface:Actuator` behaves identically to `EnergyManagementSystem:Actuator`, with the following exceptions:

- Its value is assigned by the external interface.
- Its value is fixed during the zone time step because this is the synchronization time step for the external interface.

To interface EnergyPlus using the EMS feature, the following three items are needed:

- 1) An object that instructs EnergyPlus to activate the external interface.
- 2) EnergyPlus objects that write data from the external interface to the EMS.
- 3) A configuration file to configure the data exchange.

Creating the EnergyPlus idf file

The code below shows how to set up an EnergyPlus file that uses `EnergyManagementSystem:Actuator`. To activate the external interface, we use:

```
ExternalInterface,          !- Object to activate the external interface
PtolemyServer;            !- Name of external interface
```

To declare an actuator that changes the control status of the window with name “Zn001:Wall001:Win001”, we use:

```
ExternalInterface:Actuator,
  Zn001_Wall001_Win001_Shading_Deploy_Status, !- Name
  Zn001:Wall001:Win001,          !- Actuated Component Unique Name
  Window Shading Control,        !- Actuated Component Type
  Control Status,                !- Actuated Component Control Type
  ;                               ! initial value
```

Thus, the entry is identical with `EnergyManagementSystem:Actuator`, except for the additional optional field that specifies the initial value. If unspecified, then the actuator will only be used during the time stepping, but not during the warm-up and the system sizing. Since actuators always overwrite other objects (such as a schedule), all these objects have values that are defined during the warm-up and the system sizing even if no initial value is specified.

We also want to read from EnergyPlus the outdoor temperature, the zone air temperature, the solar radiation that is incident on the window, and the fraction of time that the shading is on. Thus, we declare the output variables

```

Output:Variable,
  Environment,          !- Key Value
  Outdoor Dry Bulb,    !- Variable Name
  timestep;           !- Reporting Frequency

Output:Variable,
  *,                   !- Key Value
  Zone Mean Air Temperature, !- Variable Name
  timestep;           !- Reporting Frequency

Output:Variable,
  Zn001:Wall001:Win001, !- Key Value
  Surface Ext Solar Incident, !- Variable Name
  timestep;           !- Reporting Frequency

Output:Variable,
  *,                   !- Key Value
  Fraction of Time Shading Device Is On, !- Variable Name
  timestep;           !- Reporting Frequency

```

To specify that data should be exchanged every 10 minutes of simulation time, we enter in the idf file the section

```

Timestep,
  6;          !- Number of Timesteps per Hour

```

Creating the configuration file

Note that we have not yet specified the order of the elements in the signal vector that is exchanged between EnergyPlus and the BCVTB. This information is specified in the file variables.cfg. The file variables.cfg needs to be in the same directory as the EnergyPlus idf file. For the objects used in the section above, the file looks like

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BCVTB-variables SYSTEM "variables.dtd">
<BCVTB-variables>
  <variable source="EnergyPlus">
    <EnergyPlus name="ENVIRONMENT" type="OUTDOOR DRY BULB"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="WEST_ZONE" type="Zone Mean Air Temperature"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="Zn001:Wall001:Win001" type="Surface Ext Solar Incident"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="Zn001:Wall001:Win001" type="Fraction of Time Shading Device Is On"/>
  </variable>
  <variable source="Ptolemy">
    <EnergyPlus actuator="Zn001_Wall001_Win001_Shading_Deploy_Status" />
  </variable>
</BCVTB-variables>

```

This file specifies that the actor in the BCVTB that calls EnergyPlus has an input vector with one element that will be written to the actuator, and that it has an output vector with four elements that are computed by EnergyPlus and sent to Ptolemy. The order of the elements in each vector is determined by the order in the above XML file. Hence, the output vector that contains the signals computed by EnergyPlus has elements

```

ENVIRONMENT (OUTDOOR DRY BULB)
WEST_ZONE (Zone Mean Air Temperature)
Zn001:Wall001:Win001 (Surface Ext Solar Incident)
Zn001:Wall001:Win001 (Fraction of Time Shading Device Is On)

```

The configuration of the Ptolemy model is identical to the configuration in Example 1.

Example 3: Interface using ExternalInterface:Variable

This example implements the same controller as the Example 2. However, the interface with EnergyPlus is done using an external interface variable instead of an external interface actuator. In addition, the example uses an EnergyManagementSystem:OutputVariable to set up data that will be read by the external interface.

Similarly to EnergyManagementSystem:GlobalVariable, an ExternalInterface:Variable can be used in any EnergyManagementSystem:Program. The subject of this example is to illustrate how an ExternalInterface:Variable can be set up for use in an EnergyManagementSystem:Program. The example can be found in the BCVTB distribution in the folder examples/ePlusX-variable, where X stands for the EnergyPlus version number.

To interface EnergyPlus using an external interface variable, the following items are needed:

- An object that instructs EnergyPlus to activate the external interface.
- EnergyPlus objects that write data from the external interface to the EMS.
- A configuration file to configure the data exchange.

Creating the EnergyPlus idf file

To write data from the external interface to an EnergyPlus EMS variable, an EnergyPlus object of the following entry may be used in the idf file:

```
ExternalInterface,          !- Object to activate the external interface
  PtolemyServer;           !- Name of external interface

ExternalInterface:Variable,
  yShade,                  !- Name of Erl variable
  1;                        !- Initial value
```

During the warm-up period and the system-sizing, the variable will be set to its initial value. Afterwards, the value will be assigned from the external interface at each beginning of a zone time step and kept constant during the zone time step. From the point of view of the EMS language, ExternalInterface:Variable can be used like any global variable. Thus, it can be used within any EnergyManagementSystem:Program in the same way as an EnergyManagementSystem:GlobalVariable or an EnergyManagementSystem:Sensor.

This idf section above activates the external interface and declares a variable with name yShade that can be used in an Erl program to actuate the shading control of the window "Zn001:Wall001:Win001" as follows:

```
! EMS program. The first assignments sets the shading status and converts it into the
! EnergyPlus signal (i.e., replace 1 by 6).
! The second assignment sets yShade to
! an EnergyManagementSystem:OutputVariable
! which will be read by the external interface.
EnergyManagementSystem:Program,
  Set_Shade_Control_State,          !- Name
  Set_Shade_Signal = 6*yShade,      !- Program Line 1
  Set_Shade_Signal_01 = yShade+0.1; !- Program Line 2

! Declare an actuator to which the EnergyManagementSystem:Program will write
EnergyManagementSystem:Actuator,
  Shade_Signal, !- Name
  Zn001:Wall001:Win001, !- Actuated Component Unique Name
  Window Shading Control, !- Actuated Component Type
  Control Status; !- Actuated Component Control Type

! Declare a global variable to which the EnergyManagementSystem:Program will write
EnergyManagementSystem:GlobalVariable,
  Shade_Signal_01; !- Name of Erl variable
```

We want to read from EnergyPlus the outdoor temperature, the zone air temperature and the solar radiation that is incident on the window. Thus, we declare

```

Output:Variable,
  Environment,      !- Key Value
  Outdoor Dry Bulb, !- Variable Name
  timestep;        !- Reporting Frequency

Output:Variable,
  *,               !- Key Value
  Zone Mean Air Temperature, !- Variable Name
  timestep;        !- Reporting Frequency

Output:Variable,
  Zn001:Wall001:Win001, !- Key Value
  Surface Ext Solar Incident, !- Variable Name
  timestep;            !- Reporting Frequency

```

In addition, we want to output the variable “Erl Shading Control Status” that has been set up as

```

! Declare an output variable. This variable is equal to the shading signal + 0.1
! It will be read by the external interface to demonstrate how to receive variables.
EnergyManagementSystem:OutputVariable,
  Erl Shading Control Status, !- Name
  Shade_Signal_01,           !- EMS Variable Name
  Averaged,                  !- Type of Data in Variable
  ZoneTimeStep;              !- Update Frequency

```

To specify that data should be exchanged every 10 minutes of simulation time, enter in the idf file the section

```

Timestep,
  6;          !- Number of Timesteps per Hour

```

Creating the configuration file

Note that we have not yet specified the order of the elements in the signal vector that is exchanged between EnergyPlus and the BCVTB. This information is specified in the file variables.cfg. The file variables.cfg needs to be in the same directory as the EnergyPlus idf file. For the objects used in the section above, the file looks like

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BCVTB-variables SYSTEM "variables.dtd">
<BCVTB-variables>
  <variable source="Ptolemy">
    <EnergyPlus variable="yShade"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="ENVIRONMENT" type="OUTDOOR DRY BULB"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="WEST_ZONE" type="Zone Mean Air Temperature"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="Zn001:Wall001:Win001" type="Surface Ext Solar Incident"/>
  </variable>
  <variable source="EnergyPlus">
    <EnergyPlus name="EMS" type="Erl Shading Control Status"/>
  </variable>
</BCVTB-variables>

```

This file specifies that the actor in the BCVTB that calls EnergyPlus has an input vector with one element that will be written to the actuator, and that it has an output vector with four elements that are computed by EnergyPlus and sent to Ptolemy. The order of the elements in each vector is determined by the order in the above XML file. Note that the fourth element has the name “EMS” because it is an EnergyManagementSystem:OutputVariable. Hence, the output vector that contains the signals computed by EnergyPlus has elements

```
ENVIRONMENT (OUTDOOR DRY BULB)
WEST_ZONE (Zone Mean Air Temperature)
Zn001:Wall001:Win001 (Surface Ext Solar Incident)
EMS (Erl Shading Control Status)
```

The configuration of the Ptolemy model is identical to the configuration in the previous examples.