Documentation of ESP-r META File

Edited by Dr. Jon W. Hand
Energy Systems Research Unit
of the University of Strathclyde

Version of 20 April 2010

Introduction

ESP-r has traditionally used a distributed file store to hold the details of ESP-r models. This includes the use of different folders for different type of data e.g. zone composition is held separately from control definitions and description of networks. It also separates data for different zones into separate files. The entities in the distributed file store are flexible enough to allow users to define rooms of near-arbitrary complexity but some of the files use a syntax that is overly sensitive to the position of data entities.

The current code is implemented with a memory model which is focused on a sub-set of the model data structure depending on the current focus of the user. As other parts of the model become the focus of attention the distributed file store is scanned for the relevant data.

For models of moderate complexity most users will probably not notice the distributed file store or even view the data within the model files. The design of ESP-r is generally transparent to the users working with models of a few dozen zones.

Technical support and those who archive models will, of course, be more familiar with the internal organisation of models and the contents of files. With large models or projects which involve a number of model variants or parametric studies the above issues have become management and productivity issues for some groups.

Given the increasing use of the ESP-r simulation engine by third party applications the evolution of ESP-r is, in part, driven by the need to reduce the burden on third party developers. It is also driven by the need to support ever more complex models.

Third party developers exporting ESP-r models (or wishing to access information held within the distributed file store) must conform to the conventions of the current file store. This has proven to be problematic for some developers.

Over time an alternative file format has evolved to address some of these issues. At first this was viewed as a compact intermediate file format and a syntax was established which implemented a compact syntax. Such META entities rely on the originating application transforming their native data model entities into the compact format (using logic defined later in this document) as well as transforms and expansion of the META entities into the data model used by ESP-r.

Over time is it clear that some developers are interested in supporting the full geometric and operational complexity of the ESP-r data structure and others prefer the META (complex but compact) file format.

Thus two classes of descriptive entity are included as alternatives to the current distributed file format:

- META components which provide the essential characteristics of entities from which details are derived based on standard assumptions. Examples of this is a directive to place a window of 25% of the parent surface area into a named wall or create a box shaped zone from a few input lines.
- Analogue components which support the full complexity of the ESP-r data model. Examples of this are the use of explicit coordinates in space and surfaces tied to those coordinates.

Additional goals for the ESP-r META file:

- use a tag-data format for as many entities as possible as well as delineating related groups of data with topic start and end markers
- where the existing distributed file format is tag-data style as well as compact in form it has been retained with minimal changes

- new concepts are added to both the META file and the standard files in a similar format and with a common data structure
- much of the content of current models can be held in the META file and used to regenerate models (for sending to other users)
- an evolution of the data model to allow model files to be scanned fewer times and with more of the model details held in memory. As models evolve changes are written to file but there is little or no need to continually re-scan files.

The evolution of ESP-r is incremental and this suggests a period of transition so that the simulation engine is provided with model file store in the form it currently expects (i.e. as the model details are scanned from the META file the current distributed file format is generated. If, at a later date the underlying data structure and META file are fully evolved then it becomes an option for the ESP-r simulator to take a META file as input (perhaps in conjunction with a limited number of other files).

Maturity of the META file would be signalled by a full transit from an existing model to a META file and back with no data loss. The current META file does not yet contain the full ESP-r data model so we are some distance from the full transit.

Overview

This document provides an overview of the META file as well as an details of the API and the relationships between entities in the META file so that others can generate syntactically correct files. ESP-r may evolve more quickly than this document. If you want to explore the current MTEA file format use the export facilities of the Project Manager to export existing ESP-r models to a META file.

It is worth noting that META files generated by ESP-r use entities that support the full geometric complexity of the ESP-r data model (lets call these analogue entities) rather than the compact META entities that are interpreted and expanded as the META file is scanned. In many cases equivalent models can be defined using analogue or META approaches. The differences should be clear in the examples given below.

An example file is shown below which creates a rectangular building with a separate sloped roof space and a small pavilion on the north side via META components. The figure below shows the layout of the model to be created.

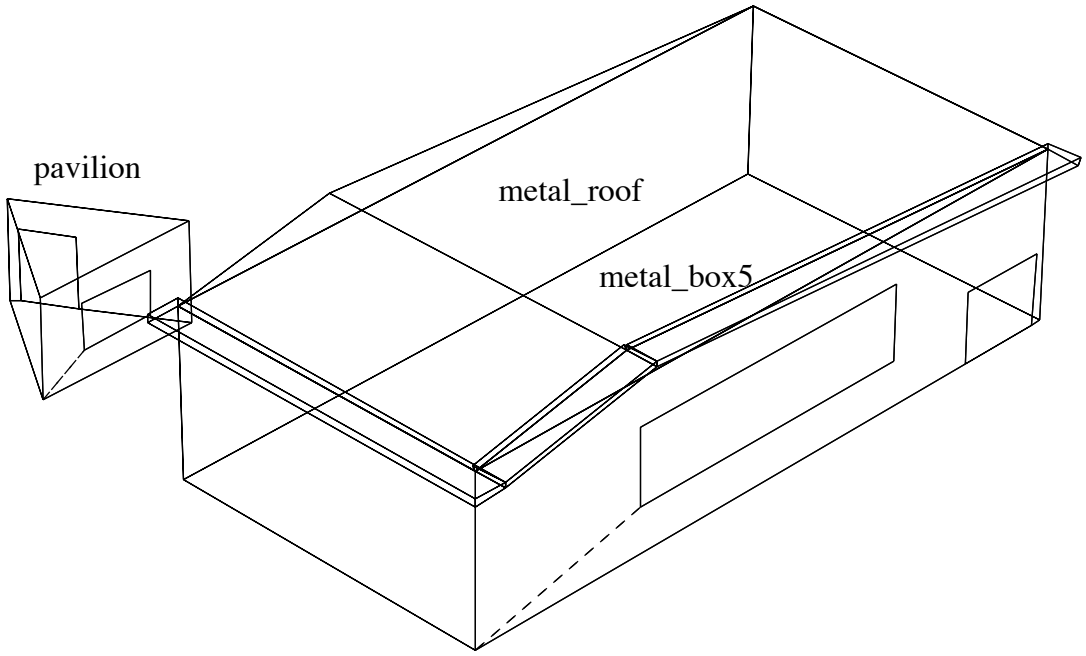

Figure 1: Overall view of model to be created

There are three types of zones included in the file â a room named metal_box5 is created from a box shape into which a window and a door is inserted. A separate triangular room has been created on the north side from a extrude shape with a window inserted into one surface and a door inserted into another. The roof space zone is created from a poly shape (it cannot be derived from a box or an extrusion).

The metal_box5 room and the triangular room use META descriptors for the general form of the zone as well as doors and windows. Such META desriptors for doors and windows are optional for poly shape zones as doors and windows are explicit surfaces in ESP-r models and the poly room type.

Combinations of these three general shape types will allow 3rd party applications to generate a wide range of ESP-r models. Any box shape can also be represented by a poly shape zone. Any extrude shape zone can also be represented via a poly shape zone. This is why META files exported by ESP-r always use the poly shape zone because it is largely insensitive to user requests to transform initail simple shapes. In some ways, use of explicit surfaces for all entities in the META file results in a more ordered file (and simpler logic in the creation of entities) than the use of META components.

Details of the box shape room META directives are shown in Figure 2, the extrude shape room META directives are shown in Figure 3 and for the poly shape room in Figure 4. These can be compared with the tokens in a standard zone geometry file in the Appendix (not yet created).

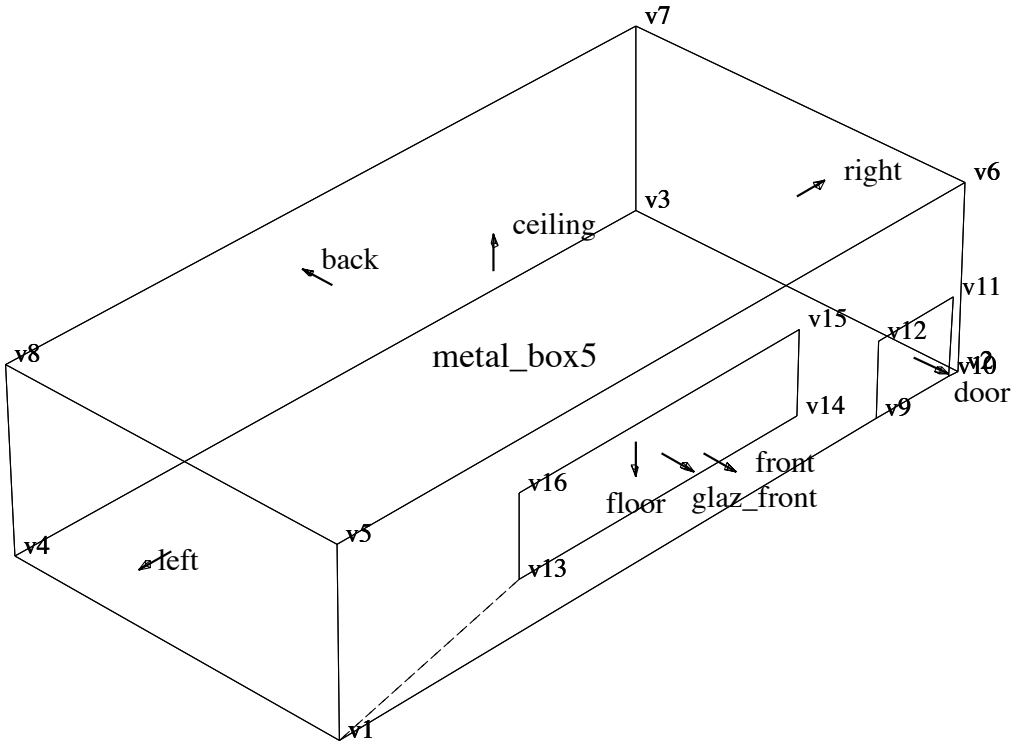

Figure 2: Details of the box shaped room

```
Zone metal_box5 ( 1) is composed of 8 surfaces and 16 vertices.
It encloses a volume of 1000.m^3 of space, with a total surface
area of 700.m^2 & approx floor area of 200.m^2
metal_box5 auto-generated for shape box & attributes.
There is 300.00m2 of exposed surface area, 300.00m2 of which is vertical.
Outside walls are 140.00 % of floor area & avg U of 0.448 & UA of 125.30
Glazing is 10.000 % of floor & 6.6666 % facade with avg U of 2.811 & UA of 56.213

A summary of the surfaces in metal_box5( 1) follows:

Sur| Area  |Azim|Elev| surface     |    geometry       | construction |environment
   | m^2   |deg |deg | name        |optical|locat| use | name         |other side
 1  74.8    180.   0. front        OPAQUE   VERT -     extern_wall  ||< external
 2  50.0     90.   0. right        OPAQUE   VERT -     extern_wall  ||< external
 3  100.      0.   0. back         OPAQUE   VERT -     extern_wall  ||< external
 4  50.0    270.   0. left         OPAQUE   VERT -     extern_wall  ||< external
 5  200.      0.  90. ceiling      OPAQUE   CEIL -     susp_ceil    ||< base:metal_roof
```

```
 6  200.      0. -90. floor        OPAQUE   FLOR -     floor_1      ||< ground profile  1
 7  5.25    180.   0. door         OPAQUE   VERT DOOR  door         ||< external
 8  20.0    180.   0. glaz_front   DCF7671_ VERT C-WIN dbl_glz      ||< external
```

The box shaped room is the most primitive initial form and as such is the least verbose of the room forms. The syntax treats doors and windows as child surfaces to the parent surfaces created with the initial room shape. Child surfaces, once created, have all of the thermophysical attributes of a surface within ESP-r.

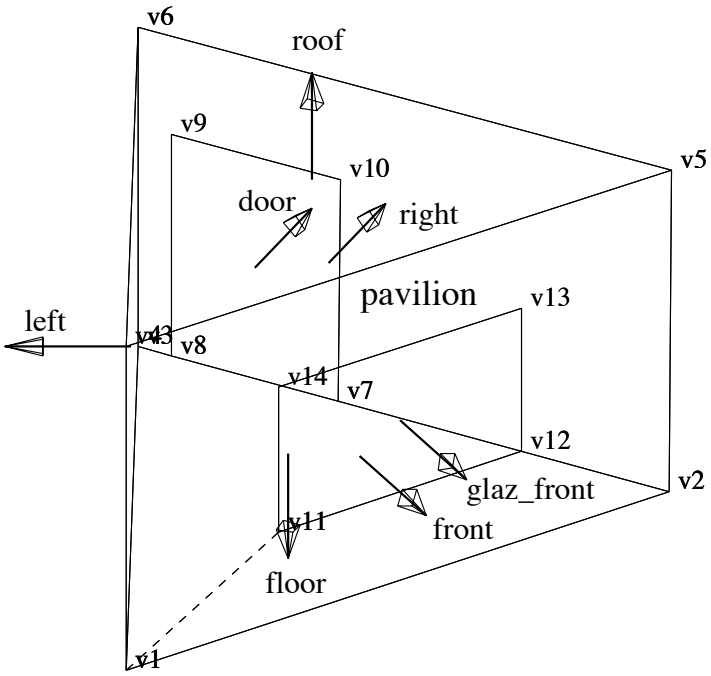

Figure 3: Details of the extrusion derived room

```
Zone pavilion ( 3) is composed of 7 surfaces and 14 vertices.
It encloses a volume of 30.0m^3 of space, with a total surface
area of 63.3m^2 & approx floor area of 10.0m^2
pavilion auto-generated for shape extrude & attributes.
There is 53.302m2 of exposed surface area, 43.302m2 of which is vertical.
Outside walls are 403.02 % of floor area & avg U of 0.621 & UA of 25.035
Flat roof is 100.00 % of floor area & avg U of 1.799 & UA of 17.992
Glazing is 30.000 % of floor & 6.9281 % facade with avg U of 2.811 & UA of 8.4319

A summary of the surfaces in pavilion( 3) follows:

Sur| Area  |Azim|Elev| surface    |    geometry       | construction |environment
   | m^2   |deg |deg | name       |optical|locat| use | name         |other side
 1  12.0    180.   0. front        OPAQUE   VERT -     extern_wall  ||< external
 2  11.0     58.   0. right        OPAQUE   VERT -     extern_wall  ||< external
 3  14.2    302.   0. left         OPAQUE   VERT -     extern_wall  ||< external
 4  10.0      0.  90. roof         OPAQUE   CEIL -     roof_1       ||< external
 5  10.0      0. -90. floor        OPAQUE   FLOR -     floor_1      ||< ground profile  1
 6  3.15     58.   0. door         OPAQUE   VERT DOOR  door         ||< external
 7  3.00    180.  -0. glaz_front   DCF7671_ VERT C-WIN dbl_glz      ||< external
```

The extrude type of room is topologically similar to the existing floor plan extrusion which users create interactively. The conventions for edge ordering (anti-clockwise looking down on the floor plate) are the same as are the limitations on the complexity of the floor plate (no more than 24 edges).

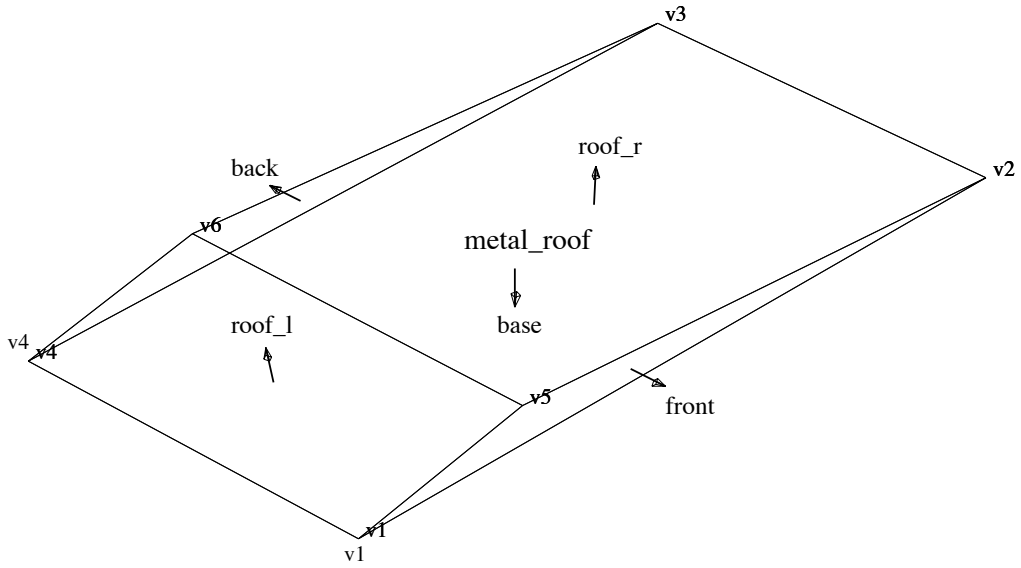

Figure 4: Details of the polygon derived room

```
Zone metal_roof ( 2) is composed of 5 surfaces and 6 vertices.
It encloses a volume of 100.m^3 of space, with a total surface
area of 421.m^2 & approx floor area of 200.m^2
metal_roof auto-generated for shape poly & attributes.
There is 221.32m2 of exposed surface area, 20.000m2 of which is vertical.
Outside walls are 10.000 % of floor area & avg U of 0.393 & UA of 7.8541
Sloped roof is 100.66 % of floor area & avg U of 1.799 & UA of 362.23

A summary of the surfaces in metal_roof( 2) follows:

Sur| Area  |Azim|Elev| surface    |    geometry       | construction |environment
   | m^2   |deg |deg | name       |optical|locat| use | name         |other side
 1  200.      0. -90. base         OPAQUE   FLOR -     susp_ceil    ||< ceiling:metal_box5
 2  10.0    180.   0. front        OPAQUE   VERT -     extern_wall  ||< external
 3  150.     90.  86. roof_r       OPAQUE   SLOP -     roof_1       ||< external
 4  10.0      0.   0. back         OPAQUE   VERT -     extern_wall  ||< external
 5  51.0    270.  79. roof_l       OPAQUE   SLOP -     roof_1       ||< external

Surfaces (all applicable) for shading analysis:
front roof_r back roof_l
No insolation analysis requested.
Block X-coord Y-coord Z-coord DX VAL. DY VAL. DZ VAL. Orien
  1     0.0    -1.0     5.0     5.1     1.0     0.2     0.0    11.3 roof_l_ovhs roof
  2     5.0    -1.0     6.0    15.0     1.0     0.2     0.0    -3.8 roof_r_ovhs roof
  3    -1.0    -1.0     5.0     1.0    11.0     0.2     0.0     0.0 roof_l_w roof
```

The poly shape room is similar in format to the native ESP-r zone geometry file and it is able to represent any box or extrude initial shape as well as enclosures of arbitrary complexity. It is governed by the conventions of the GEN shape zone in ESP-r. If a room cannot be created from an initial box or extrude shape then the poly shape type can be used. In the example the intent is a roof space with a sloped roof because neither of them includes a provision for sloped surfaces.

Since a GEN description in ESP-r includes explicit polygons for doors and windows they can be so described in the META file. And the language tokens for adding child windows and doors are also applicable for a poly shaped room. The choice of explicit polygons or polygons generated from META instructions, or even a mix of the two is up to the user. General advise is not to mix both styles within a single parent surface (it is possible for the parsing code to mis-understand mixed styles).

The input file that generated the above model is listed below. After the file contents each of the tokens will be explained.

```
*silent_input
*menu,Creates three zone model in /tmp/poly3
*doc,Model of a box and roof and triangual room within distributed folders to be created in /tmp/poly3 and th
*action,new,poly,distributed,/tmp/poly3
*zonpth ../zones # path to zones
*netpth ../nets # path to networks
*ctlpth ../ctl # path to controls
*radpth ../rad # path to radiance files
*imgpth ../images # path to project images
*tmppth ../tmp # path to project scratch folder
*docpth ../doc # path to project documents
*dbspth ../dbs # path to local databases
*weather_station,uk_oban
*site_loc,45.320,0.000,0.0,6 # latitude, long diff, time zone
*site_exp,1,28.0,38.0,34  # site expoure views to sky ground buildings
*ground_refl_annual,0.20
# model of ground followed by monthly ground reflectance values
*ground_refl_monthly,2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.7
# days with snow (matching ground albido model 2
*monthly_snow_days,29,27,22,4,0,0,0,0,0,4,20
*monthly_profile,1
5.00,4.00,3.50,3.60,6.60,10.20,13.00,15.00,15.00,13.00,10.00,7.00
# start of zone information
*start_zone,metal_box5
*shape,box
*origin,0.0,0.0,0.0  # origin of zone
*size,20.0,10.0,5.0  # length width height (m)
*rotate,0.0,0.0,0.0     # pending rotation angle X Y
*previous_rotate,10.0,0.0,1.0  # prior rotation angle X Y
*glaze,1,1,20.0         # how many, index of parent and percent
*door,1,1,2.5    # how many, index of parent and width of door
*surface,front,extern_wall,0,0,0
*surface,right,extern_wall,0,0,0
*surface,back,extern_wall,0,0,0
*surface,left,extern_wall,0,0,0
*surface,ceiling,susp_ceil,3,2,1
*surface,floor,floor_1,4,1,0
*surface,door,door,0,0,0
*surface,glaz_front,dbl_glz,0,0,0
# shading directives
*shad_calc,none  # no temporal shading requested
# insolation directives
*insol_calc,none  # no insolation requested
*usage,pattern,all,allcas,nothing_in_room.opr   # casual gain pattern
*environment,comvective_night_off  # name of zone control pattern
*heating,18.0,0.0   # setpoint occupied & unoccupied
*cooling,26.0,100.0 # setpoint occupied & unoccupied
*end_zone
*start_zone,metal_roof
*shape,poly
*nbwalls,5
*cord,0.0,0.0,5.0
*cord,20.0,0.0,5.0
*cord,20.0,10.0,5.0
*cord,0.0,10.0,5.0
*cord,5.0,0.0,6.0
*cord,5.0,10.0,6.0
*origin,0.0,3.0,0.0  # base and topsilent and ignored
*size,0.0,0.0,0.0    # not used for extrude shape.
*previous_rotate,10.0,0.0,1.0  # prior rotation angle X Y
*surface,base,susp_ceil,3,1,5
*list,4,1,4,3,2
*surface,front,extern_wall,0,0,0
*list,3,1,2,5
*surface,roof_r,roof_1,0,0,0
*list,4,2,3,6,5
*surface,back,extern_wall,0,0,0
*list,3,3,4,6
*surface,roof_l,roof_1,0,0,0
*list,4,4,1,5,6
# shading directives
*shad_calc,all_applicable   4 # list of surfs
  2  3  4  5
# insolation directives
*insol_calc,none  # no insolation requested
```

```
#  *obs = solar obstructions
*solar_grid,20 20 # solar gridding density
*obs3,0.000,-1.000,5.000,5.099,1.000,0.200,0.000,11.310,0.000,roof_l_ovhs,roof  # block  1
*obs3,5.000,-1.000,6.000,15.033,1.000,0.200,0.000,-3.814,0.000,roof_r_ovhs,roof  # block  2
*obs,-1.000,-1.000,5.000,1.000,11.000,0.200,0.000,roof_l_w,roof  # block  3
*usage,pattern,all,allcas,nothing_in_room.opr   # casual gain pattern
*environment,comvective_night_off  # name of zone control pattern
*heating,18.0,0.0   # setpoint occupied & unoccupied
*cooling,26.0,100.0 # setpoint occupied & unoccupied
*end_zone
*start_zone,pavilion
*shape,extrude
*nbwalls,3
*cord,0.0,15.0
*cord,5.0,15.0
*cord,2.5,19.0
*origin,0.0,3.0,0.0  # base and top and ignored
*size,0.0,0.0,0.0    # not used for extrude shape.
*previous_rotate,10.0,0.0,1.0  # prior rotation angle X Y
*glaze,1,1,20.0         # how many, index of parent and percent
*door,1,2,1.5    # how many, index of parent and width of door
*surface,front,extern_wall,0,0,0
*surface,right,extern_wall,0,0,0
*surface,left,extern_wall,0,0,0
*surface,roof,roof_1,0,0,0
*surface,floor,floor_1,4,1,0
*surface,door,door,0,0,0
*surface,glaz_front,dbl_glz,0,0,0
# shading directives
*shad_calc,none  # no temporal shading requested
# insolation directives
*insol_calc,none  # no insolation requested
*usage,pattern,all,allcas,nothing_in_room.opr   # casual gain pattern
*environment,comvective_night_off  # name of zone control pattern
*heating,18.0,0.0   # setpoint occupied & unoccupied
*cooling,26.0,100.0 # setpoint occupied & unoccupied
*end_zone
*end
```

The tokens are listed below. Where there are start and end marks these are discussed together. Tokens are almost always followed by a comma and a comma is typically used to separate data. Unless specified, the order of the tokens is not critical although the hierarchy is suggested to be from site related to building related and then to zone related.

```
*silent_input
 . . .
*end
```

The *silent_input is the first tag in the file and marks its type and *end is the last tag in the file and marks the end of the model description.

```
*menu,Creates two zone model in /tmp/poly3
```

or

```
*title,Creates two zone model in /tmp/poly3
```

The *menu (depreciate) or *title line is followed by a block of text (up to 72 char) which is used as the title of the model within ESP-r as well as text that could be used by applications which might scan a number of META files and present them for user selection. The *title line is typically found near the header of the file.

```
*doc,Model of a box and roof with distributed folders to be
  created in /tmp/poly3 and then set site to use data from
  irl_dublin_iwec climate file.
```

the *doc line is followed by a block of text (up to 248 char) on one line which provides a summary of the model.

Model folder related data

```
*action,new,poly,distributed,/tmp/poly3
```

The *action line gives directives as to how the model is to be created. The 2nd token is either:

- new - create a new model in a specified folder
- within - create a new model within the current folder

The 3rd token is the root name of the model. In the above case the model files will be poly.cfg and poly.cnn etc.

The 4th token is either:

- distributed - create cfg dbs nets tmp zones folders
- single - all the model files are in one folder.

The 5th token is the absolute path to the location of the model. In the above case the model files will be /tmp/poly3/cfg /tmp/poly3/zones etc.

```
*zonpth ../zones # path to zones
*netpth ../nets # path to networks
*ctlpth ../ctl # path to controls
*radpth ../rad # path to radiance files
*imgpth ../images # path to project images
*tmppth ../tmp # path to project scratch folder
*docpth ../doc # path to project documents
*dbspth ../dbs # path to local databases
```

There are an optional set of tokens which define the folder layout of the ESP-r model derived from the META file. For the case where the 4th token of the *action line is distributed the standard ESP-r folder structure is created e.g. ../zones ../nets ../ctl if no directives are given. If a bespoke folder structure is required, for example to place local databases in ../databases instead of ../dbs then a *dbspth line would be added.

In each case there are two tokens on the line. The first is the tag and the second is a relative path to the relevant folder.

When the project manager of ESP-r exports an existing model these lines are added to ensure that the equivalent folder structure is recovered.

Database related data

ESP-r models have associated databases. If the meta file contains no directives then the current default databases are assumed to be used. There are directives that identify materials, constructions, optics, pressure distributions, event profiles and miscel components. The tags used identify whether the file is local to the model, located via an explicit path or held in the standard ESP-r databases folder.

Databases that exist outside of the model, for example in a users folder such as /home/fred/databases or in the standard ESP-r distribtion such as /usr/esru/esp-r/databases or C:\Esru\esp-r\databases will be scanned in as the model is created. If databases are located within the model folder structure then it cannot be assumed that they will be available as ESP-r parses the META file and creates the folders. This can lead to unknown constructions and an inability to create the zone construction files. However, newer *surface entries can hold the name of the surface optical attribute as well as the construction attribute and it is possible to build more of the model zone files.

Examples are shown below:

```
*mat ../dbs/hotel.materialdb  # materials database
*mlc ../dbs/hotel.constrdb    # constructions database
*opt ../dbs/hotel.opticdb     # optical db
*prs ../dbs/hotel.pressuredb  # pressure coef db
*evn /home/fred/databases/profiles.db1 # event profiles db
*mscldb /home/fred/databases/mscomp.db1 # miscel components db
*pdb /usr/esru/esp-r/databases/plantc.db1 # plant template db
```

or

```
*stdmat material.db3.a # standard materials database
*stdmlc multicon.db3   # standard constructions database
*stdopt optics.db2     # standard optical db
*stdprs pressc.db1     # standard pressure coef db
*stdevn profiles.db2.a # standard event profiles db
*stdpdb plantc.db1     # plant template db
```

The *mat entry (for example) is followed by ../dbs is a relative path from the model configuration file. The *mscldb entry is an example of an absolute path. These tags are also used in the model configuration file.

The *stdmat entry (for example) is followed by a file name (but no path). ESP-r executables have a compiled-in assumption as to where standard databases are located and a *stdmat *stdmlc *stdopt *stdprs *stdevn or *stdpdb tag tells ESP-r to pre-pend the path to the supplied file name. This allows META files and the models they contain to be less sensitive to where ESP-r is installed and where models are held.

Site related data

```
*weather_station,uk_oban
```

The *weather_station line includes one token which is the name of an ESP-r climate file. The file is assumed to exist in the standard location of climate files for the installed version of ESP-r. The file name can be up to 32 characters (since no path information is included). The *weather_station line is optional. If it is not included then the default assumptions for climate file will be used.

```
*site,45.320,0.000,0.0,1 # latitude, long diff, time zone, exposure index
```

or

```
*site_loc,45.320,0.000,0.0  # latitude, long diff, time zone
*site_exp,1,0.36,0.36,0.28  # site expoure index views to sky ground buildings
```

The *site line includes 4-7 tokens as follows: 1st is latitude (degrees), 2nd is the difference in degrees between the site and the time meridian for the time zone, 3rd is the hours before or after GTM, 4th is an index of the site exposure (same as in the ESP-r configuration file). If the index is equal to 8 then there are three more tokens on the line to represent the view to the sky, ground and other buildings. The *site line is optional. If it is not included, default assumptions apply. This tag has been superceeded by *site_loc and *site_exp.

The *site_loc line is a newer format which holds the latitude, longitude difference and hours from time zone.

The *site_exp line is a newer format which holds the index of the site exposure followed by the views from building surfaces to the sky, ground and other buildings.

```
*ground_refl_annual,0.20
```

The *ground_refl_annual line has one token â the reflection from the ground which is assumed to apply all months of the year. This line is optional and defaults will be used if it is not included.

```
# model of ground followed by monthly ground reflectance values
*ground_refl_monthly,2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.7
# days with snow (matching ground albido model 2
*monthly_snow_days,29,27,22,4,0,0,0,0,0,4,20
```

The *ground_refl_monthly line specifies which method is to be used for treating the ground where snowfall is a significant issue. The 1st token is a number which specifies the index of the method. If the value is 2 then a *monthly_snow_days line is expected to occur in the file. If the value is 3 then a *hourly_snow_depth line is expected. lThe remainder f the *ground_refl_monthly line are a dozen reflection values.

The *monthly_snow_days line includes a dozen tokens of number of days in each month that have snow on the ground.

The *hourly_snow_depth line has one token which is the name of the file which holds the hourly data for snow depths.

```
*monthly_profile,1
5.00,4.00,3.50,3.60,6.60,10.20,13.00,15.00,15.00,13.00,10.00,7.00
```

The *monthly_profile line is included when user defined monthly ground temperature profiles are required. There is one token on the line specifying the number of sets of temperature profiles. This is followed by (one line each) a dozen values of the temperature at each month. If there are three sets specified then there will be three lines of data expected.

Assessment related data

ESP-r has the concept of simulation parameter sets which record user preferences for how many simulations to run, their periods and what results files should be generated. There are two tags associated with this topic as follows:

```
*sim_parameters,3,4,1,6
```

The *sim_parameters line has 5 tokens: the 1st is the number of startup days, the 2nd is the number of zone-side timesteps (between 1 and 60), the 3rd is the number of plant timesteps per zone-side timestep (between 1 and 100), the 4th is the save level (between 0 and 6).

The *assessments line can take several forms as shown below.

```
*assessments,annual

*assessments,winter,typical_week
*assessments,winter,fortnight
*assessments,winter,season

*assessments,spring,typical_week
*assessments,spring,fortnight
*assessments,spring,season

*assessments,summer,typical_week
*assessments,summer,fortnight
*assessments,summer,season

*assessments,autumn,typical_week
*assessments,autumn,fortnight
*assessments,autumn,season

*assessments,three_season,typical_week
*assessments,three_season,fortnight
*assessments,three_season,season

*assessments,five_season,typical_week
*assessments,five_season,fortnight
*assessments,five_season,season
```

The 1st token defines how many assessments are defined. The allowed entries are:

- annual - self explanatory, no further tokens
- single - one assessment, followed by details of the period (NOT YET DONE)
- winter - one winter assessment (based on the climate file definition of the winter season), followed by tokens specifying the period
- spring - one spring assessment (based on climate definition), followed by tokens specifying the period
- summer - one summer assessment (based on climate definition), followed by tokens specifying the period
- autumn - one autumn assessment (based on climate definition), followed by tokens specifying the period
- three_season - as in IPV winter/transition/summer assessments, followed by tokens specifying the periods
- five_season - as in IPV winter (jan)/spring/summer/autumn/winter (dec) assessments, followed by tokens specifying the periods

The 2nd token specifies the type of period. The allowed entries are:

- typical_week - if the climatelist file includes a week that is typical of the season then it will be used.
- fortnight - if the climatelist file includes a week that is typical of the season then a fortnight will be defined beginning 3 days before and 4 days after the typical week.
- season - if the climatelist file includes a definition of a season then the assessment will include all of the days of the season.

Currently META files exported from ESP-r do not yet include the *assessments data line. This will be included at a later date.

The *ipv is just a placeholder for future generation of ipv related data. This is an optional item. Syntax example:

```
*ipv
. . . details to be worked out
```

One third party tool that uses the META file required a syntax for reporting on high level performance issues. The approach taken was to read a series of topics and generate an XML file which includes the topics in a report. This facility is work in progress so expect the syntax to change over time.

The *start_xml and *end_xml items are used to define the boundaries for the output results that are written in the generated input.xml file. The description starts with *start_xml and every output parameter is written to the next lines (one parameter per line) until it finds the *end_xml line. The simulation will generate an out.csv (as long as xml is active and as long as save level 5 is used) that contains the outputs between the *start_xml and *end_xml. Syntax example:

```
*start_xml
building/all_zones/supplied_energy/heating
building/all_zones/supplied_energy/cooling
building/all_zones/envelope/all_components/heat_loss
building/all_zones/envelope/all_components/heat_gain
building/all_zones/envelope/infiltration/heat_gain
building/all_zones/envelope/infiltration/heat_loss
building/all_zones/internal_gains/total
building/all_zones/insolation/total
building/all_zones/insolation/adverse
building/all_zones/insolation/useful
*end_xml
```

Zone related data

```
*start_zone,metal_box5
 . . .
*end_zone
```

Data related to a thermal zone are held between the *start_zone and *end_zone tags. The 2nd token in the *start_zone line is the name of the zone (up to 12 char with no spaces).

```
*shape,box

*shape,extrude

*shape,poly
```

The *shape line hould follow the *start_zone line. The 2nd token can be:

- box - a rectangular zone (origin, length, width, height)
- extrude - a zone extruded from a floor plan (a set of X and Y)
- poly - a zone composed of polygons (points and edges)

Depending on the shape type a series of lines will follow. Some tags are applicable to specific shapes and others are used with all shapes.

Use of the box shape

There are a sequence of data lines which are typical in a zone which is defined as a **box shape**.

```
*start_zone,metal_box5
*shape,box
*origin,0.0,0.0,0.0  # origin of zone
*size,20.0,10.0,5.0  # length width height (m)
*rotate,0.0,0.0,0.0  # pending rotation angle X Y
*previous_rotate,10.0,0.0,1.0  # prior rotation angle X Y
. . .
```

An *origin line for shape type **box** has three tokens. The tokens are real numbers representing the X Y Z coordinates in metres of the origin of the zone. If the shape type is 'poly' no *origin line is expected, but if found none of the tokens are used.

The *size line is only used with shape type **box**. There are three real number tokens, 1st is the length (m) along the east axis if there is no rotation, 2nd is the depth (m) along the north axis if there is no rotation, 3rd is the height (m). If a *size line is encountered for other shape types the data is ignored.

A *rotate or *previous_rotate line can happen anywhere within the *start_zone and *end_zone tags. It has three real data which are the degrees (positive is anti-clockwise) of rotation to be applied to the zone followed by the X and Y coordinate to rotate around. A *rotate or *previous_rotate token is optional for **box** and **extrude** and **poly** shape types.

A *rotate tag signals a pending operation while the *previous_rotate tag records the users prior request for rotation (for a possible un-do command). Third party application can use the *rotate tag to apply transforms to an initial set of coordinates.

Use of extrude shape type

A zone which is derived from a floor plan extrusion follows the rules for a floor plan extrusion geometry file within ESP-r. The sequence of data lines follows the following pattern:

```
*start_zone,pavilion
*shape,extrude
*nbwalls,3
*origin,0.0,3.0,0.0  # base and top and ignored
*cord,0.0,15.0
*cord,5.0,15.0
*cord,2.5,19.0
. . .
```

For an **extrude** shape zone a *nbwalls line follows the *shape line and precedes the details of the geometry. The 2nd token is the number of walls in the zone (i.e. the floor and ceiling and inserted glazing or doors are not included in this number).

An *origin line for shape type **extrude** has three tokens. The 1st token is the Z at the base of the zone, the 2nd token is the Z at the top of the zone and the 3rd token is ignored. found none of the tokens are used.

For an **extrude** shape zone one or more *cord lines are expected. There are two data tokens which represent the X and Y co-ordinates in metres. The number of *cord lines should match the number of walls. The order of the *cord lines should trace the edges of the floor surface on which the extrusion is to be built in an anti-clockwise direction looking down at the floor.

Use of poly shape type

The **poly** shape is the most general form of zone description and is composed of a number of coordinates in space followed by surface definitions which are defined as a list of edges which define arbitrary polygons. A poly shape type can be used to explicitly represent the full geometric complexity supported by ESP-r. If you have a sloped ceiling or a window which is not rectangular or wish to include complex surfaces floating within the zone to represent thermal mass this is the shape type to use.

```
*start_zone,metal_roof
```

```
*shape,poly
*nbwalls,5
*cord,0.0,0.0,5.0
*cord,20.0,0.0,5.0
. . .
*surface,base,susp_ceil,3,1,5
*list,4,1,4,3,2
. . .
```

The syntax of *nbwalls is the same for a zone of shape **poly** but represents the number of explicit polygons in the room.

For a **poly** shape zone one or more *cord lines are expected. There are three data tokens which represent the X, Y and Z co-ordinates in metres. There should be one *cord line for each vertex used in the zone. All vertices should be unique points in space. If two *cord lines are identical then it is likely that the topology rules of ESP-r will fail. The ordering of the *cord lines should be such that the first *cord line is assumed to represent the 1st vertex and the 2nd *cord line is the 2nd vertex.

For a **poly** shaped zone surface polygons make reference to the *cord lines. The syntax includes a *list line after each *surface line as follows:

```
*surface,base,susp_ceil,3,1,5
*list,4,1,4,3,2
```

The *surface line is discussed below. The *list line starts with an integer representing the number of vertices which are used to make up the edges of the polygon. This is followed by the index of each *cord line (implied in the ordering of the *cord lines). Note: *list lines should not be used with **box** or **extrude** shapes.

Again, a poly shaped room with all surfaces explicitly represented may be easier to generate and less prone to error than box and extrude shapes (even though more lines of data are used).

```
*surface,front,extern_wall,0,0,0
```

Each surface line has a number of data. The 1st is the name of the surface (up to 12 char with no spaces), the 2nd is the name of the construction (up to 12 char with no spaces). The name of the construction MUST match a name in the construction database associated with the model.

The last three data in the *surface line are integers that define the boundary conditions at the other face of the surface. The values use the same order and syntax as are found in the model configuration connections file. The 1st is the type of connection and the 2nd and 3rd provide supplemental information.

For example *surface,ceiling,susp_ceil,3,2,1 stipulates that the ceiling in the zone metal_box5 is thermally connected to the 2nd zone in the model (metal_roof) and the 1st surface in that zone (base). If you look closely in the *surface line for base in metal_roof you will see that the indicies point back to the ceiling in metal_box5. This pairing of suface boundary conditions is required by all modules of ESP-r.

Essentially the agent creating the META file is assumed to know the 'topology' of the model and to follow the descriptive rules which apply to an ESP-r model *.cnn file. One way to check the logic of a third party application as it generates a META file is to create an equivalent model in ESP-r and export a META file and look for differences.

Data in *surface lines are used for zones of all shape types. The number of *surface lines is equivalent to the number of parent surfaces created in by the initial shape plus one for each glazing and door if those META directives have been included. For example, a **box** shape zone will have at least six *surface lines. An **extrude** shape zone will have a *surface line for each of the *nbwalls. A **poly** shaped room will have a *surface line for each of the *nbwalls stipulated.

The order of *surface lines is critical and begins with the parent surfaces and their implicit order (discussed above). After the parent surfaces have been created any *door directives are used. In the example file above the box shaped zone has a door inserted in the first (front) surface and thus it will be the 7th surface in the zone. After *door directives have been acted on *glazing directives are processed. In the box shaped room there is one glazing and it becomes the 8th surface in the zone.

META components

Several of the components in the META file are high level entities that are interpreted and expanded into the analogue components that ESP-r understands.

```
*glaze,1,1,20.0          # how many, index of parent and percent
```

A *glaze line is a **META** component which provides instructions for inserting transparent surfaces into the parent surfaces of the zone (the ones created via the initial **box** or **extrude** shape command). If there are no windows to be inserted as child surfaces then the *glaze line should be omitted. A *glaze line is optional for shape type poly. When ESP-r exports META files it does not use *glaze lines.

The number of data items in the *glaze line will vary depending on how many windows are to be added.

The 1st data is an integer specifying the number of glazing META objects to be inserted into the zone. For EACH glazing there are two data: 1st is the index of the parent surface. The order of the creation of parent surfaces is specific to the zone shape type. For a box shape (with no rotation) the order is front, right, back, left, top, base. For an extruded shape the walls are created based on set of supplied co-ordinates followed by the top and then the base (e.g. a four wall extrusion is identical to a box shape).

For a poly shape zone the index of the parent surface follows from the order of the *surface lines. Care is required in specifying the parent surface index in poly shaped zones!

The 2nd glazing data item is the percentage of the parent surface area to give to the glazing. The glazing is placed as a rectangle near the centre of the parent surface. If this is not specific enough then the glazing must be represented as an explicit polygon within a poly shaped zone.

```
*door,1,1,2.5    # how many, index of parent and width of door
```

A *door line is a **META** component which provides instructions for inserting a door into a parent surface in the zone (the ones created via the initial **box** or **extrude** shape command). It follows a similar pattern to the *glaze described above except that the 2nd door data item is the width of the door (m). The current convention is that doors are inserted near the right edge of the parent surface (looking from the outside). The height of the door is assumed to be 2.1m if the overall height of the parent surface is greater than 2.3m otherwise the height of the door is 0.3m less than the height of the parent surface. As with glazing, if you want to be specific about the location and size of a door use a poly room type and represent the door as an explicit surface. A *door line is optional for shape type poly. When ESP-r exports META files it does not use *door lines.

Often users have a need to represent internal mass within a zone such as desks and filing cabinets or to efficiently add back-to-back surfaces within a zone to represent explicit blinds. While a 'poly' shaped zone could include such surfaces explicitly, the 'box' and 'extrude' zone type syntax does not directly support such entities. All zone shapes can include one or more *mass lines. The *mass META directive creates a pair of horizontal or vertical rectangular surfaces (via origin orientation and length/width directives) which are connected to each other as partitions (type 3). All the surface attributes are included in the *mass line and thus no *surface entry is required. *mass entries are represented as explicit surfaces in any META files exported from ESP-r.

```
*mass,VM,0.1,5.0,0.1,9.3,2.3,180.0,mass_l,door
```

Where the 2nd token is either VM (vertical mass) or HM (horizontal mass). The 3rd token is X origin (m), the 4th token is Y origin (m), the 5th token is Z origin (m), the 6th token is length (m) and the 7th token is height (for vertical) or width (for horizontal). The 8th token is the orientation of the surface normal (0.0 is North, 180.0 is south). The 9th token is the name of the first surface to be created (the matched surface has a '_' appended. The last token is the construction to be used.

An example of the use of *mass entries can be seen below. A box shape has been used for the overall enclosure and three *mass directives have created internal partitions (one long and two short lengths). This allows the simpler box shape to be used instead of a poly shape.

```
<< to be done >>
```

Figure 5: Inclusion of internal partitions via *mass directive.

Directives for increasing model resolution

ESP-r models include facilities to increase the resolution of assessments and the META file includes some of these directives.

```
# shading directives
*shad_calc,none  # no temporal shading requested
# insolation directives
*insol_calc,none  # no insolation requested
```

ESP-r supports the calculation of shading patterns on exterior surfaces at each hour of the day for a typical day in each month of the simulation as well as the distribution of direct solar radiation within zones (insolation). Shading calculations can be undertaken if there are shading obstructions defined in the model (see the next section).

Whether or not shading patterns have been calculated the user can request that the pattern of solar insolation is calculated. There are thus separate tokens in the META file which provide directives as to how shading and insolation should be treated. In the above example the directive token **none** stipulates that the calculation is not requested and that the default assumptions for shading and/or insolation is to be used within the simulation.

```
# shading directives
*shad_calc,all_applicable   4 # list of surfs
 2  3  4  5
# insolation directives
*insol_calc,all_applicable  1  # list of surfaces
 7
```

In the above example the token is **all_applicable** and this is followed by an integer (the number of surfaces which are to be considered for shading). The next line contains the list of surface indices for which shading should be calculated (e.g. 2 is the second surface in the zone). Surfaces in this list must face the **outside**.

In the above *insol_calc line the **all_applicable** has a similar syntax. In this case it identifies **transparent surfaces which face the outside** and is followed by a line which includes the index of this surface.

```
*obs,-1.00,-1.00,5.00,1.00,11.00,0.20,0.00,roof_l_w,roof  # block  3
```

Solar obstructions are defined via one or more *obs lines anywhere between the *start_zone and *end_zone tags. There are 9 data tokens as follows:

- X origin of the block (m)
- Y origin of the block (m)
- Z origin of the block (m)
- length (m) along east axis if no rotation
- depth (m) along north axis if no rotation
- height (m) along Z axis
- degrees of rotation (positive is anticlockwise) about the origin.
- name of the block (up to 12 char)
- name of the construction of the block (up to 12 char) and must point to an existing construction within the construction database used by the model.

In the example above there is a block which is 1m long and 11m wide and 20cm tall with an orgin at -1.0 -1.0 5.0 which is not rotated (its length runs along the X axis).

```
*obs3,0.00,-1.00,5.00,5.099,1.00,0.20,0.00,11.310,0.00,roof_l_ovhs,roof  # block  1
```

Solar obstructions which can be rotated in multiple axis are defined via one or more *obs3 lines anywhere between the *start_zone and *end_zone tags. There are 11 data tokens. The first 6 are identical to the *obs type. The remaining tokens are as follows:

- degrees of rotation (positive is anticlockwise) about the origin.

- degrees of elevation (positive is upwards) between the ground plane and the length of the block.
- degrees of tile (positive is upwards) between the ground plane and the width of the block. This rotation is currently NOT implemented.
- name of the block (up to 12 char)
- name of the construction of the block (up to 12 char) and must point to an existing construction within the construction database used by the model.

In the example above, there is a block which is 5.099m long, 1.0m wide and 0.2m high which has an origin at XYZ 0.0 -1.0 5.0 and which is tilted upwards along its length by 11.31 degrees.

Non-geometric zone attributes

Thermal simulation models include a number of attributes to zones related to how rooms are used (occupants, lighting, small power) as well as the environmental systems associated with zones. The META file includes tokens which can be used to define such attributes. The current approach points to pre-defined room usage patterns as discussed below. In a future version detailed schedules will be supported. Currently, META files exported from ESP-r do not include entries for *usage.

```
*usage,pattern,all,allcas,nothing_in_room.opr   # casual gain pattern
```

The *usage line provides directives about what is happening within a zone. The 2nd token is 'pattern' and this directs ESp-r to look in the standard pattern folder (esp-r/training/pattern) for the installed version of ESP-r. The 3rd token is either:

- all - import infiltration schedules and control
- infil - import infiltration schedules

The 4th token is:

- allcas - import all casual gain schedules in the pattern file.
- - - skip casual gains schedules in the pattern file.

The 5th token is the name of the pattern file to scan (up to 32 char).

```
*environment,convective_night_off  # name of zone control pattern
```

An *environment line gives directives about the control to be applied to the current zone. This is work in progress and nothing is yet done with the information. The data tokens have not yet been finalised.

```
*heating,18.0,0.0   # setpoint occupied & unoccupied
*cooling,26.0,0.0   # setpoint occupied & unoccupied
```

A *heating and *cooling line give directives about heating and cooling setpoints for the current zone. This is work in progress and it is only used for an ideal control when token *ideal_control exists. The data tokens have not yet been finalised.

```
*ideal_control
```

When *ideal_control is included together with tokens *heating and *cooling, an ideal control is created for the current zone. It uses a zone air temperature sensor and actuator, one day type for the whole year, 24 hours per day and unlimited heating and cooling power. The setpoints are taken from the first values after *heating and *cooling.

META files with explicit components

It is possible to create a META file using only analogue components rather than META components. As an example of this, the initial model presented in Figure 1 has been exported from ESP-r using all **poly** shapes and explicit surfaces. Compare this with the **box** and **extrude** descriptions above. Note that the *heating and *cooling and *ideal_control directives are currently not attributed in an exported META file.

```
*silent_input
*title,Creates a three zone model from META descriptions
*doc,Creates a three zone model from META descriptions
*date,Wed Oct  7 11:58:47 2009  # latest file modification
```

```
*action,new,box_ext_poly4,distributed,box_ext_poly4
*weather_station,XXX  # climate file
*site_loc,52.000, 0.000, 0.00 # latitude deg long-diff deg time-zone GMT
*site_exp, 1, 0.36, 0.36, 0.28 # exposure-index (std) sky ground building views
*ground_refl_annual, 0.200 # ground reflection (avg for year)
*start_zone,metal_box5    # zone name
*shape,poly      # polygon enclosure
*cord,0.0000,0.0000,0.0000  # X Y Z for   1
*cord,20.0000,0.0000,0.0000  # X Y Z for   2
*cord,20.0000,10.0000,0.0000  # X Y Z for   3
*cord,0.0000,10.0000,0.0000  # X Y Z for   4
*cord,0.0000,0.0000,5.0000  # X Y Z for   5
*cord,20.0000,0.0000,5.0000  # X Y Z for   6
*cord,20.0000,10.0000,5.0000  # X Y Z for   7
*cord,0.0000,10.0000,5.0000  # X Y Z for   8
*cord,17.2000,0.0000,0.0000  # X Y Z for   9
*cord,19.7000,0.0000,0.0000  # X Y Z for  10
*cord,19.7000,0.0000,2.1000  # X Y Z for  11
*cord,17.2000,0.0000,2.1000  # X Y Z for  12
*cord,5.5279,0.0000,1.3820  # X Y Z for  13
*cord,14.4721,0.0000,1.3820  # X Y Z for  14
*cord,14.4721,0.0000,3.6180  # X Y Z for  15
*cord,5.5279,0.0000,3.6180  # X Y Z for  16
*rotate,  0.0000  # rotation around zone origin
*nbwalls,  8  # number of surfaces
*surface,front,extern_wall,0,0,0  # surface attributes
*list,14,1,9,12,11,10,2,6,5,1,13,16,15,14,13  #  1
*surface,right,extern_wall,0,0,0  # surface attributes
*list,4,2,3,7,6  #  2
*surface,back,extern_wall,0,0,0  # surface attributes
*list,4,3,4,8,7  #  3
*surface,left,extern_wall,0,0,0  # surface attributes
*list,4,4,1,5,8  #  4
*surface,ceiling,susp_ceil,3,2,1  # surface attributes
*list,4,5,6,7,8  #  5
*surface,floor,floor_1,4,1,0  # surface attributes
*list,6,1,4,3,2,10,9  #  6
*surface,door,door,0,0,0  # surface attributes
*list,4,9,10,11,12  #  7
*surface,glaz_front,dbl_glz,0,0,0  # surface attributes
*list,4,13,14,15,16  #  8
# shading directives
*shad_calc,none  # no temporal shading requested
# insolation directives
*insol_calc,none  # no insolation requested
*heating,  0.00  # ideal heating stepoint
*cooling,  0.00  # ideal heating stepoint
*ideal_control,  0  # ideal control linkage
*end_zone
*start_zone,metal_roof    # zone name
*shape,poly      # polygon enclosure
*cord,0.0000,0.0000,5.0000  # X Y Z for   1
*cord,20.0000,0.0000,5.0000  # X Y Z for   2
*cord,20.0000,10.0000,5.0000  # X Y Z for   3
*cord,0.0000,10.0000,5.0000  # X Y Z for   4
*cord,5.0000,0.0000,6.0000  # X Y Z for   5
*cord,5.0000,10.0000,6.0000  # X Y Z for   6
*rotate,  0.0000  # rotation around zone origin
*nbwalls,  5  # number of surfaces
*surface,base,susp_ceil,3,1,5  # surface attributes
*list,4,1,4,3,2  #  1
*surface,front,extern_wall,0,0,0  # surface attributes
*list,3,1,2,5  #  2
*surface,roof_r,roof_1,0,0,0  # surface attributes
*list,4,2,3,6,5  #  3
*surface,back,extern_wall,0,0,0  # surface attributes
*list,3,3,4,6  #  4
*surface,roof_l,roof_1,0,0,0  # surface attributes
*list,4,4,1,5,6  #  5
# shading directives
*shad_calc,all_applicable   4 # list of surfs
  2  3  4  5
# insolation directives
*insol_calc,none  # no insolation requested
#  *obs = solar obstructions
*solar_grid,20 20 # solar gridding density
*obs3,0.000,-1.000,5.000,5.099,1.000,0.200,0.000,11.310,0.000,roof_l_ovhs,roof  # block  1
*obs3,5.000,-1.000,6.000,15.033,1.000,0.200,0.000,-3.814,0.000,roof_r_ovhs,roof  # block  2
```

```
*obs,-1.000,-1.000,5.000,1.000,11.000,0.200,0.000,roof_l_w,roof  # block  3
*heating,  0.00  # ideal heating stepoint
*cooling,  0.00  # ideal heating stepoint
*ideal_control,  0  # ideal control linkage
*end_zone
*start_zone,pavilion      # zone name
*shape,poly      # polygon enclosure
*cord,0.0000,15.0000,0.0000  # X Y Z for   1
*cord,5.0000,15.0000,0.0000  # X Y Z for   2
*cord,2.5000,19.0000,0.0000  # X Y Z for   3
*cord,0.0000,15.0000,3.0000  # X Y Z for   4
*cord,5.0000,15.0000,3.0000  # X Y Z for   5
*cord,2.5000,19.0000,3.0000  # X Y Z for   6
*cord,3.4540,17.4736,0.0000  # X Y Z for   7
*cord,2.6590,18.7456,0.0000  # X Y Z for   8
*cord,2.6590,18.7456,2.1000  # X Y Z for   9
*cord,3.4540,17.4736,2.1000  # X Y Z for  10
*cord,1.3820,15.0000,0.8292  # X Y Z for  11
*cord,3.6180,15.0000,0.8292  # X Y Z for  12
*cord,3.6180,15.0000,2.1708  # X Y Z for  13
*cord,1.3820,15.0000,2.1708  # X Y Z for  14
*rotate,  0.0000  # rotation around zone origin
*nbwalls,  7  # number of surfaces
*surface,front,extern_wall,0,0,0  # surface attributes
*list,10,1,2,5,4,1,11,14,13,12,11  #  1
*surface,right,extern_wall,0,0,0  # surface attributes
*list,8,2,7,10,9,8,3,6,5  #  2
*surface,left,extern_wall,0,0,0  # surface attributes
*list,4,3,1,4,6  #  3
*surface,roof,roof_1,0,0,0  # surface attributes
*list,3,4,5,6  #  4
*surface,floor,floor_1,4,1,0  # surface attributes
*list,5,1,3,8,7,2  #  5
*surface,door,door,0,0,0  # surface attributes
*list,4,7,8,9,10  #  6
*surface,glaz_front,dbl_glz,0,0,0  # surface attributes
*list,4,11,12,13,14  #  7
# shading directives
*shad_calc,none  # no temporal shading requested
# insolation directives
*insol_calc,none  # no insolation requested
*heating,  0.00  # ideal heating stepoint
*cooling,  0.00  # ideal heating stepoint
*ideal_control,  0  # ideal control linkage
*end_zone
*end
```

Revisions

The following is a summary of changes made to the META file format. This section is work-in-progress.

March 2010 - added *zonpth, *netpth, *ctlpth, *radpth, *imgpth, *tmppth, *docpth and *dbspth tokens to support custom model folder naming. Relax 10 zone limit on model complexity.

February 2010 - added *previous_rotate token to record prior rotation requests and modified the *rotate token to include the X and Y of the point for zone rotation.

December 2009 updated documentation about META file.

September 2009 - modified the *site token to *site_loc and *site_exp tokens, removed the *action,cont option to simplify the code and allowed longer text block after the *menu token.

September 2009 - added dual rotation shading obstruction descriptions *obs3.

August 2009 - added directives for shading and insolation calculations.

June 2009 - added directives for the day-type callendar equivalent to that used in the ESP-r configuration file.

March 2009 - added facility to export a META file from an existing ESP-r model in the prj module.