



Extensibility Tools and API User Guide

DesignBuilder v2025.1

August 2025

Contents

Contents	1
Overview	2
The DesignBuilder API	2
DB.Api.Environment	2
Object Attributes	2
DesignBuilder Hooks	3
Extending DesignBuilder With Scripts	4
What is a Script?	4
Writing a C# Script	5
Writing a Python Script	7
Importing Other Scripts Into a Script	9
Extending DesignBuilder with Plugins	10
What is a Plugin?	10
Writing a Plugin	10
Writing a Plugin With a Dynamic Menu	15
Plugin API Versions	21
IPlugin supported hook points	21
IPlugin2 supported hook points	21
Working with the DesignBuilder API	23
DesignBuilder Building Model	23
Decomposition Level Object Data and Attributes	24
The Site Object, Model Status and the API Environment Class	25
DesignBuilder Tables	26
Component and Template database tables	27
Categories	28
Results tables	28
Navigating Model Objects via the API	28
Zone Surface Types and Holes	29
Extracting Geometry	30
Setting Startup Location	31
API Reference	32
Example Scripts and Plugin Code Repository	32
Appendix 1 List of Available Tables	32
Columns	37
Appendix 2 - DesignBuilder Dialogs and Database Tables	38
DesignBuilder Data (.dat) file	38
Header	39
Data	39
Table of Tables	39
DesignBuilder Table Format Files	40
DesignBuilder Edit Screen Format file	45
Model Attributes	49
Attribute List	49

Extensibility Tools User Guide

Overview

DesignBuilder can be extended through scripts or plugins. Each system is targeted at a different use case and complexity level. Both systems may be used at the same time, and multiple active plugins and scripts are supported.

Scripts are intended to hook into certain stages of a DesignBuilder simulation in order to modify or report on the process. Plugins are a superset of Scripts, providing extra functionality such as the ability to create a custom sub-menu in DesignBuilder's main menu, which allows for ad-hoc processing of DesignBuilder data.

Scripts can be written in C# or Python. Plugins are written in the C# programming language.

The DesignBuilder API

The [DesignBuilder API](#) can be accessed through all of DesignBuilder scripting and plugin development options, including Python and C# scripting and plugins.

DB.Api.Environment

For C#, DesignBuilder exposes its API to both extensibility systems by setting the `ApiEnvironment` property on the `IProperties` interface, which `IScript` and `IPlugin` inherit. The `Api.Environment` object is the root API object, which can be used to access the rest of the API.

For Python, the global variable `api_environment` is set.

During start-up, DesignBuilder instantiates a single `DB.Api.Environment` object. This singleton object is used by all active scripts and plugins, and as such, a script or plugin cannot rely on the state of the system at any given time. For example, suppose a script modifies the system at point A in a simulation. The script is then called again at point B and may expect the state of the system to be as it was at point A. However, it is possible that another script or plugin (or even DesignBuilder itself) also modified the system between point A and B. We therefore suggest that no assertions are made about the state of the system at any given point.

`DB.Api.Environment` is partitioned into several major areas, including:

- CFD operations, accessible via `DB.Api.Environment.CfdOperations`, which exposes functionality relating to DesignBuilder's CFD system
- Event subscriptions, accessible via `DB.Api.Environment.Events`, which allow for scripts and plugins to be notified of certain changes to the system
- HVAC operations, accessible via `DB.Api.Environment.HvacOperations`, which exposes functionality relating to DesignBuilder's HVAC system
- Site operations, accessible via `DB.Api.Environment.Site`, which exposes functionality relating to the active site. Functionality relating to Buildings, Blocks, Zones, Surfaces, etc can be accessed from the `Site` object
- Environment operations, accessible directly from `DB.Api.Environment`, which exposes a lot of miscellaneous functionality such as undo/redo, navigation, etc.

Object Attributes

DesignBuilder uses object attributes to store much of the data for any given object (such as a building, block, zone, etc). An attribute is simply a key-value pair of strings that can be accessed via the `GetAttribute` and

SetAttribute methods on any object that supports attributes. For example, to get the occupancy density for a zone you would use

DB.Api.Zone.GetAttribute("OccupancyValue"). To set the occupancy density to 0.1 you would use DB.Api.Zone.SetAttribute("OccupancyValue", (0.1).ToString()).

Generally, an object can have a lot of attributes and it is impractical to list and describe each one. Nevertheless, to get full use of the API you will need to know an attribute's name, or key, so that you can access or change its value. To help with this, you can enable DesignBuilder to show an attribute's name in a tooltip in DesignBuilder's GUI. To do this, simply enable Tools > Program options > Interface > Interface Style > Show attribute names in tooltips (you will need to enable "Show Tooltips in Model data" first, if it is not already enabled).

With "Show attribute names in tooltips" enabled, you can hover over an attribute in DesignBuilder and the tooltip will display that attribute's name, which you can then use with GetAttribute and SetAttribute. For example:



DesignBuilder Hooks

During a DesignBuilder simulation, DesignBuilder can pass control to a script or plugin at certain points in the process. These points are called hook points, or hooks, or call points. Both scripts and plugins have access to the same hooks and any implemented hooks are called for each active script and plugin. That is to say, if you have 2 active plugins and 3 active scripts, all of which implement the same hook, DesignBuilder will pass control to all 5 hooks in the following order: All scripts in the order they appear in the Script Manager, followed by all plugins in alphabetical order of the plugin's assembly name.

The following hooks are available, with each hook point (in a category) being called in the order they are written:

Simulation:

- **BeforeEnergyIdfGeneration (C#)/before_energy_idf_generation (Python)** is called immediately before the EnergyPlus IDF file is created
- **BeforeEnergySimulation (C#)/before_energy_simulation (Python)** is called immediately before an EnergyPlus simulation starts
- **AfterEnergySimulation (C#)/after_energy_simulation (Python)** is called immediately after an EnergyPlus simulation has finished
- **BeforeHeatingIdfGeneration (C#)/before_heating_idf_generation (Python)** is called immediately before the EnergyPlus IDF file is created for a Heating design simulation
- **BeforeHeatingSimulation (C#)/before_heating_simulation (Python)** is called immediately before a Heating design simulation starts
- **AfterHeatingSimulation (C#)/after_heating_simulation (Python)** is called immediately after a Heating design simulation has finished
- **BeforeCoolingIdfGeneration (C#)/before_cooling_idf_generation (Python)** is called immediately before the EnergyPlus IDF file is created for a Cooling design simulation
- **BeforeCoolingSimulation (C#)/before_cooling_simulation (Python)** is called immediately before a Cooling design simulation starts
- **AfterCoolingSimulation (C#)/after_cooling_simulation (Python)** is called immediately after a Cooling design simulation has finished

Optimisation:

- **BeforeOptimisationStudy (C#)/before_optimization (Python)** is called immediately before an optimisation study starts
- **AfterOptimisationStudy (C#)/after_optimisation (Python)** is called immediately after an optimisation study has finished

- **OnDesignVariableChanged (C#)/on_design_variable_changed (Python)** is called when an optimization design variable of type “Custom Script” is changed before an optimisation simulation. The id and value of the variable are passed to this hook point. The Id’s start at 10000 and are ordered as they appear in the Design Variables window
- Note that optimisation will run multiple simulations and for each simulation the **BeforeEnergyIdfGeneration** and **BeforeEnergySimulation** hooks are called

CFD:

- **BeforeCfdSimulation (C#)/before_cfd_simulation (Python)** is called immediately before a CFD simulation starts
- **AfterCfdSimulation (C#)/after_cfd_simulation (Python)** is called immediately after a CFD simulation has finished

Daylight Simulation:

- **BeforeDaylightSimulation (C#)/before_daylight_simulation (Python)** is called immediately before a daylighting simulation starts
- **AfterDaylightSimulation (C#)/after_daylight_simulation (Python)** is called immediately after a daylighting simulation has finished

Cost and Carbon:

- **BeforeCostAndCarbon (C#)/before_cost_and_carbon (Python)** is called immediately before generating a cost and carbon report
- **AfterCostAndCarbon (C#)/after_cost_and_carbon (Python)** is called immediately after generating a cost and carbon report

Misc:

- **AtCommandLine (C#)/at_command_line (Python)** is called when DesignBuilder is run from the command-line with the argument /process=ExternalCommand_<arg>, where arg is an optional string that is passed to the hook point
- **ScreenChanged (C# plugin v2 only)** is called when DesignBuilder changes screen. An enum corresponding to the new screen is passed as an argument.
- **ModelLoaded (C# plugin v2 only)** is called after a model (dsb file) has been opened or a new model has been created.
- **ModelUnloaded (C# plugin v2 only)** is called after a model has been unloaded (the file has been closed).

Note that before each hook point is called¹, for C#, the ActiveBuilding property on IProperties will be set to the instance of the building that the hook point is called by. For Python, the active_building global variable will be set. In the case that a building is not currently selected, ActiveBuilding/active_building will be null.

See section [Plugin API Versions](#) for more information on the available plugin versions.

Extending DesignBuilder With Scripts

What is a Script?

A script is a means for users to extend a DesignBuilder simulation by providing custom functionality via the C# or Python programming languages. A script may modify or report on DesignBuilder’s internal data at each of the hook points described in section

DesignBuilder Hooks.

¹ Except for ModelLoaded and ModelUnloaded

Unlike a plugin, a script may only interact with DesignBuilder at the defined hook points and only exist during the lifetime of a simulation.

Scripts are saved with a model in the .dsb file, unlike plugins, which are separate from a model.

Writing a C# Script

To extend DesignBuilder with a script using the C# programming language:

1. Open DesignBuilder, open or create a model and navigate to Tools > Scripts. This will open the *Script Manager*.
2. In the Script Manager, ensure the *Enable scripts* option is checked and click on the *Script* sub-item, which will open the *Select the Script* dialog. Here, you can see a list of scripts that are supplied with DesignBuilder. You will also notice that DesignBuilder supports three types of scripts; CS-Script, EMS, and Python-Script

EMS scripts do not support the DesignBuilder API and work differently from CS-Scripts and Python-Scripts. Documentation for EMS scripts can be found [here](#).

Python-Scripts are not fully supported in DesignBuilder v6.0. Access to the API will be provided for Python scripts in v6.2.

3. In the Select the Script dialog, open the CS-Script tree folder and select any existing script. From here you may create a new script, or copy or edit the selected script by using the icons at the bottom of the dialog.

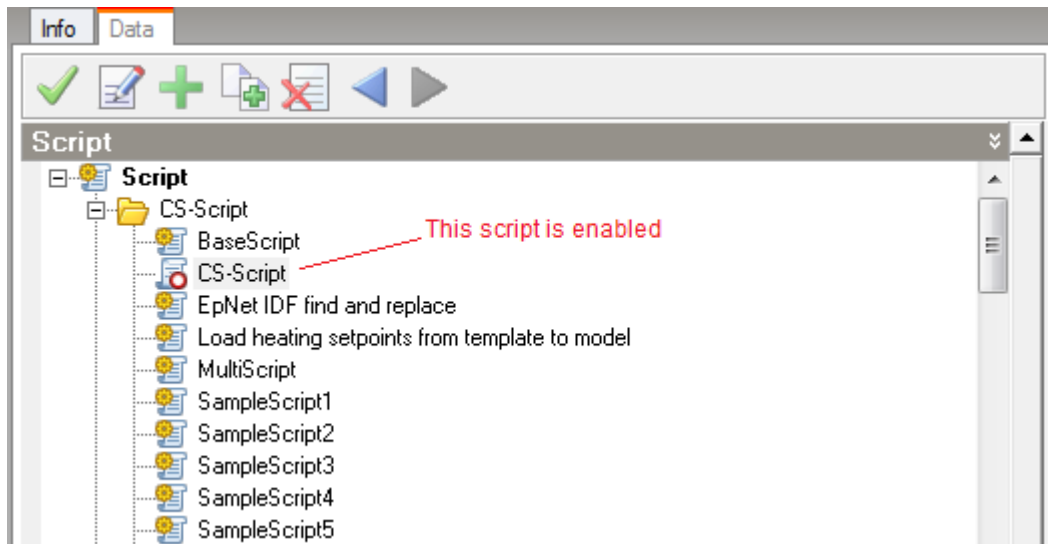
Press the  button to create a new script.

4. The *Edit Script* window provides a basic text editor into which you can write your code. It is suggested, however, that an editor more suited to C# development is used to write your script from which you can copy the code into the Edit Script text box.

Generally, a script will consist of a single class. You may choose to scope your class to a namespace, but it is not necessary. Copy the following code into the Edit Script text box

```
public class ExampleScript
{
}
```

5. Click on the *Compile script* button in the Info pane and you should see the message “Script compiled with no errors.” Press the OK button on the message box. The Compile script button is a convenient way of checking your script is syntactically correct before we run a simulation.
6. In order to enable this script to run during a simulation, make sure the *Enable program* checkbox in the Edit Script window is checked. It can be found just above the Edit Script textbox under the *Script* header.
7. Press the OK button on the Edit Script window. This will take us back to the Script Manager window. You should now see your script listed in the tree view of the data pane with a red circle on its icon. This indicates that the script is enabled and will be run during a simulation.



8. Press the OK button on the Script Manager window. Navigate to the Simulation tab in DesignBuilder and run a new simulation. The simulation should complete as normal.

This is the basic process required to create, write, and enable a script in DesignBuilder. The script we wrote in step 4, however, doesn't do a whole lot. In order for the script to be more useful it needs to hook in to DesignBuilder through various entry points.

9. Navigate back to the Script Manager and click on the script created in step 4. To edit the script, click the

Edit highlighted item button , which will take you back to the Edit Script window, where you can alter the script's code.

10. In order for DesignBuilder to pass control of a simulation to a script, a script must expose at least one of the hook points described in section
11. *DesignBuilder Hooks*. Each hook point defines a different stage in a simulation. A script can expose as many hook points as it chooses.

From a C# point-of-view, a script must implement the `DB.Extensibility.Contracts.IScript` interface. This interface declares a method for each hook point that DesignBuilder can call.

Because it's likely that a script will not want to provide an implementation for every hook point, a base class (`DB.Extensibility.Contracts.ScriptBase`) is also provided, which implements default functionality for the `IScript` interface.

Add the following code to the existing script:

```
using DB.Extensibility.Contracts;

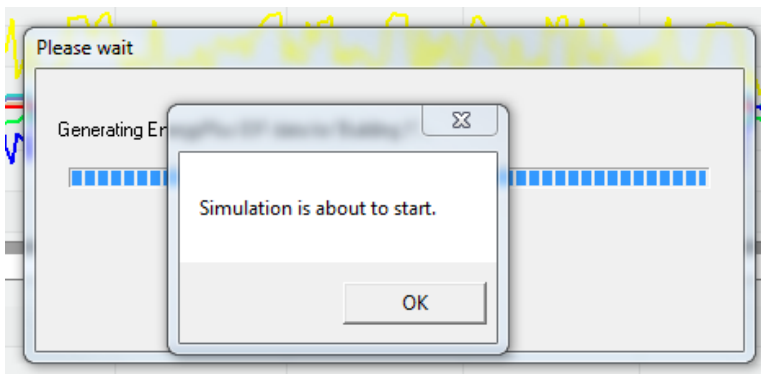
public class ExampleScript : ScriptBase, IScript
{
}
```

12. The script now does a bit more than the previous version, but not much. Technically, it exposes all hook points to DesignBuilder, but each is implemented with the default behaviour provided by `DB.Extensibility.Contracts.ScriptBase`. The default behaviour of `ScriptBase` is to do nothing. To remedy that, the script will have to override one of the methods. For the purposes of this example override the `BeforeSimulation` hook point as follows

```
using System.Windows.Forms;
using DB.Extensibility.Contracts;

public class ExampleScript : ScriptBase, IScript
{
    public override void BeforeEnergySimulation()
    {
        MessageBox.Show("Simulation is about to start.");
    }
}
```

- Click the Compile script button to make sure the code is syntactically correct, then press the OK button on the Edit Script window. Now press the OK button on the Script Manager window and run a new simulation. After the Loading data and Generating IDF stages of simulation, a messages box should appear as depicted below



To continue with the simulation press the OK button on the message box.

- The script is now a fully functioning script that displays a message before simulation starts. It's not hard to imagine that this script could be modified to open a more useful window that allows users to view, or modify certain options before continuing with the simulation.

For more ideas of what can be done with a script please examine the example scripts provided by default. They can be found in the Script Manager window.

Writing a Python Script

DesignBuilder Python scripting is based on the IronPython implementation of the Python v3.4 programming language.

To extend DesignBuilder with a script using the Python programming language:

- Open DesignBuilder, open or create a model and navigate to Tools > Scripts. This will open the *Script Manager*.
- In the Script Manager, ensure the *Enable scripts* option is checked and click on the *Script* sub-item, which will open the *Select the Script* dialog. Here, you can see a list of scripts that are supplied with DesignBuilder. You will also notice that DesignBuilder supports three types of scripts; CS-Script, EMS, and Python-Script.
- In the Select the Script dialog, open the Python-Script tree folder and select any existing script. From here you may create a new script, or copy or edit the selected script by using the icons at the bottom of the dialog.

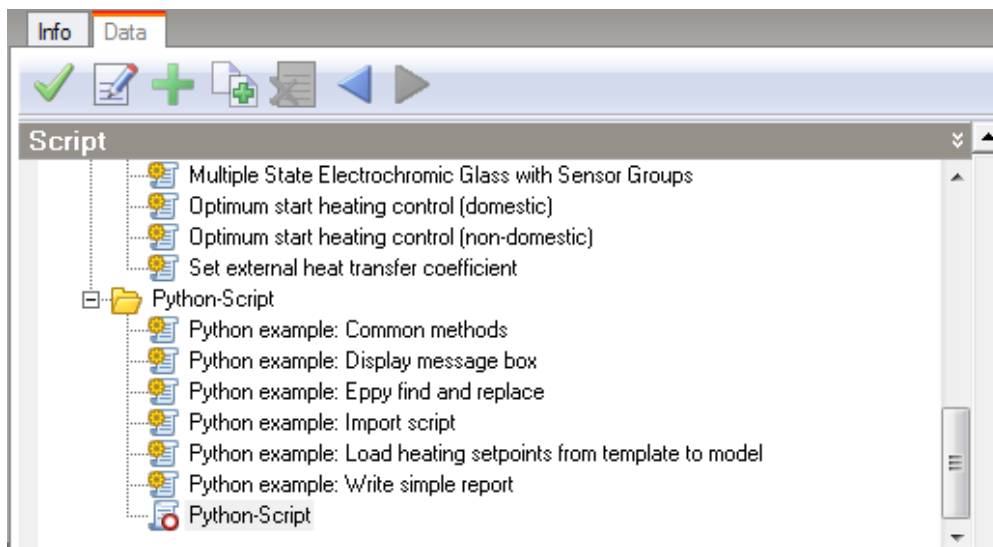
Press the  button to create a new script.

- The *Edit Script* window provides a basic text editor into which you can write your code. It is suggested, however, that an editor more suited to Python development is used to write your script from which you can copy the code into the Edit Script text box.

In order to enable a script to run during a simulation, make sure the *Enable program* checkbox in the Edit Script window is checked. It can be found just above the Edit Script textbox under the *Script* header.

The *Compile script* button in the Info pane is a convenient way of checking your script is syntactically correct before we run a simulation.

- Press the OK button on the Edit Script window. This will take us back to the Script Manager window. You should now see your script listed in the tree view of the data pane with a red circle on its icon. This indicates that the script is enabled and will be run during a simulation.



- Press the OK button on the Script Manager window. Navigate to the Simulation tab in DesignBuilder and run a new simulation. The simulation should complete as normal.

This is the basic process required to create, write, and enable a script in DesignBuilder. Our script, however, is empty and doesn't do a whole lot. In order for the script to be more useful it needs to hook in to DesignBuilder through various entry points.

- Navigate back to the Script Manager and click on the script created in step 4. To edit the script, click the

Edit highlighted item button , which will take you back to the Edit Script window, where you can alter the script's code.

- In order for DesignBuilder to pass control of a simulation to a script, a script must expose at least one of the hook points described in section
- DesignBuilder Hooks*. Each hook point defines a different stage in a simulation. A script can expose as many hook points as it chooses.

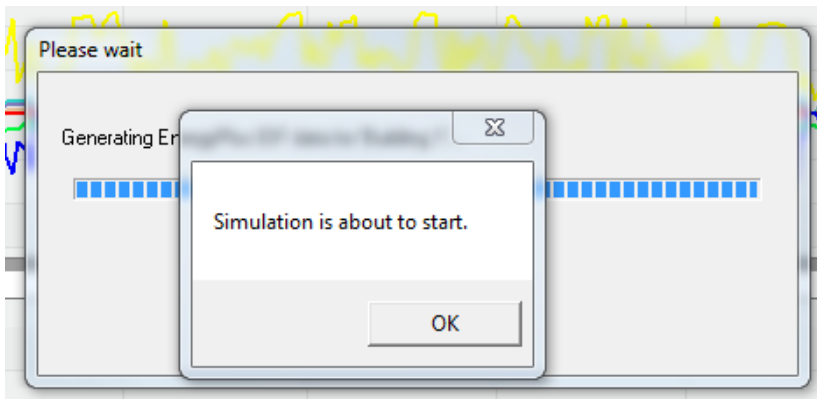
From a Python point-of-view, all that is required is for a script to define a function with the same name as one of the hook points.

Add the following code to the existing script:

```
import ctypes

def before_energy_simulation():
    ctypes.windll.user32.MessageBoxW(
        0, "Simulation is about to start.", "Title", 0)
```

10. Click the Compile script button to make sure the code is syntactically correct, then press the OK button on the Edit Script window. Now press the OK button on the Script Manager window and run a new simulation. After the Loading data and Generating IDF stages of simulation, a messages box should appear as depicted below



To continue with the simulation press the OK button on the message box.

11. The script is now a fully functioning script that displays a message before simulation starts. It's not hard to imagine that this script could be modified to open a more useful window that allows users to view, or modify certain options before continuing with the simulation.

For more ideas of what can be done with a script please examine the example scripts provided by default. They can be found in the Script Manager window.

Importing Other Scripts Into a Script

A DesignBuilder script can import 1 or more other scripts before it is run. This is useful if you want to combine scripts that may expose different hook points as if it were a single script, or if you want to store utility classes/methods in some base script, which you can use in other scripts.

In order to import a C# script into another, simply use the directive `css_import <other script's name>` as a comment at the top of the importing script. For example, say "ScriptParent" intends to import functionality from "ScriptChild", ScriptParent would include `//css_import ScriptChild` at the top of its file (this can be before or after the using statements). For Python, use the from/import directives. For example, to import a method called "show_message" from "ScriptChild", ScriptParent would include `from ScriptChild import show_message`

It is important to note that both the parent script and all child scripts must be enabled in the Script Manager window.

The name of the script is the one that appears in the Script Manager window with the following alterations: Any spaces or instances of the characters '<' (less than), '>' (greater than), '\' (back slash), '/' (forward slash), '?' (question mark), ':' (colon), ';' (semi-colon), '"' (quotation mark), ',' (comma), '*' (asterisks) will be replaced by underscores.

For example, a script with the name "Example: My script", will need to be imported with the name "Example__My_script".

If more than one enabled script implements the same hook point, each one will be executed in alphabetical order of the class name that implements it (C#), or the filename that implements it (Python).

Extending DesignBuilder with Plugins

What is a Plugin?

A plugin provides a more complete way of extending DesignBuilder. Plugins are a superset of C# Scripts so they are able to take advantage of the same hook points that scripts can use with the addition of being able to add a menu to DesignBuilder's top-level menu (if they choose).

Unlike scripts, which are interpreted at runtime, plugins are pre-built assemblies that are loaded by DesignBuilder. This gives the developer more freedom than a script because they are able to reference other assemblies and nuget packages.

Plugins can be used:

- As a more complete scripting solution. By implementing only the hook points and not providing a menu structure, a plugin will not be visible to a user but will still be used by DesignBuilder during a simulation. This is useful if you want to reference third-party assemblies, or your own libraries in your solution
- To allow for ad-hoc calculations, reports, or modifications to DesignBuilder's data. Because plugins can create their own menu in DesignBuilder they do not rely on being called by the hook points like scripts do. Plugin operations can be called whenever a menu item is pressed
- A more complete extension of DesignBuilder. Because plugins are virtually unrestricted they may provide a complete GUI-based solution. Whilst they cannot integrate with DesignBuilder's GUI, they can launch their own windows and work in a very similar manner to DesignBuilder's own Model Data Grid View (which can be found in Tools > Model data grid view... when the Edit tab is active)

One downside of the plugin system, however, is that **plugins are not saved with the .dsb file**. If you therefore want to share a model and a plugin is vital to your work, you will also need to share the plugin assemblies. Scripts on the other hand are saved with the model.

Writing a Plugin

DesignBuilder loads plugins at start-up from .NET assemblies. Therefore, to develop a plugin you must create a new .NET class library. It is assumed that you already know how to do this as it is out of the scope of DesignBuilder's help.

Note that the API has only been tested using .NET framework v4.6 and the C# programming language. You should therefore use this combination for your plugin development work.

The example provided here assumes a .NET class library using the C# programming language.

To extend DesignBuilder with a plugin:

1. Communication between DesignBuilder and a plugin is achieved through the DesignBuilder API and a plugin-specific interface. It is therefore important that your project assembly references the DB.Api and DB.Extensibility.Contracts assemblies.

The DB.Api assembly can be found in the "Components/DB.Api" directory of DesignBuilder's installation directory. DB.Extensibility.Contracts can be found in the "Components/DB.Extensibility" directory.

2. DesignBuilder uses Microsoft's Managed Extensibility Framework (MEF) to locate and load plugins. Your project assembly must therefore also reference System.ComponentModel.Composition
3. Now that your project assembly has the correct references it's time to create a class that DesignBuilder can interact with. As mentioned, DesignBuilder communicates with a plugin via a plugin-specific interface, namely DB.Extensibility.Contracts.IPlugin.

Create the following new class in your project:

```

using System;
using DB.Extensibility.Contracts;

namespace DB.Extensibility.Plugins
{
    public class ExamplePlugin : PluginBase, IPlugin
    {
        public bool HasMenu
        {
            get { throw new NotImplementedException(); }
        }

        public string MenuLayout
        {
            get { throw new NotImplementedException(); }
        }

        public bool IsMenuItemVisible(string key)
        {
            throw new NotImplementedException();
        }

        public bool IsMenuItemEnabled(string key)
        {
            throw new NotImplementedException();
        }

        public void OnMenuItemPressed(string key)
        {
            throw new NotImplementedException();
        }

        public void Create()
        {
            throw new NotImplementedException();
        }
    }
}

```

Here, ExamplePlugin inherits IPlugin and PluginBase, both of which are provided by DB.Extensibility.Contracts. IPlugin is the interface that DesignBuilder uses to communicate with a plugin and declares methods for plugin creation/initialisation, methods for the description and interaction of a plugin's menu, and methods relating to the various simulation hook points as described in section

4. *DesignBuilder* Hooks.

PluginBase provides default implementations of the simulation hook points. Inheriting from PluginBase is not required, but it's useful if don't intend to use the simulation hook points.

Note: The choice of namespace and class names are arbitrary.

5. DesignBuilder uses MEF to locate and load plugins. To make your plugin MEF aware, add the following code:

```
using System;
using System.ComponentModel.Composition;
using DB.Extensibility.Contracts;

namespace DB.Extensibility.Plugins
{
    [Export(typeof(IPlugin))]
    public class ExamplePlugin : PluginBase, IPlugin
    {
        public bool HasMenu
        {
            get { throw new NotImplementedException(); }
        }
        ...
    }
}
```

If you are unfamiliar with MEF, the new code tells MEF that the ExamplePlugin class should be made available to an importer as something that implements IPlugin.

6. The plugin is now properly configured with the required references, implements the required interface, provides default implementations for each simulation hook point, and is publically available to DesignBuilder for import. For all intents and purposes it is a valid plugin that can be loaded and used by DesignBuilder.

Nevertheless, as it's currently implemented, the plugin is not very useful. To make it more useful we'll create a menu structure that will appear in DesignBuilder's top-level menu. The menu will have only 1 item, which will display a message box when pressed.

A plugin's menu structure is described by a custom description language that is returned to DesignBuilder as a string via IPlugin.MenuLayout. The description language has the following rules:

- The '*' (asterisk) character indicates a new menu item
- Zero or more '>' characters indicate the indentation level of the menu item
- The ',' (comma) character indicates the split between the menu item's name as will be displayed in DesignBuilder and the item's id (or key), which will be used in communication between DesignBuilder and the plugin. The text preceding the ',' (and after any '*' or '>' characters) is the displayed name, and the text proceeding the ',' is the menu item's id

Example 1: The string `"*Test Plugin,root*>Item 1,i1*>>Sub-item A,1a"` would create a menu structure that looks like:

```
Test Plugin    (key = root)
-- Item 1      (key = i1)
-- Sub-item A  (key = 1a)
```

If a user press Sub-item A, DesignBuilder would inform the plugin by passing key "1a" to IPlugin.OnMenuItemPressed.

Example 2: The string `"*Test Plugin,root*>Item 1,i1*>Item 2,i2"` would create a menu structure that looks like:

```
Test Plugin (key = root)
-- Item 1   (key = i1)
-- Item 2   (key = i2)
```

As well as MenuLayout, IPlugin declares 4 other methods that are used in the creation and usage of a plugin's menu. They are:

- **HasMenu**, which simply returns whether the plugin has a menu or not. If this returns false, the other menu-related methods will not be called
- **IsMenuItemVisible** returns whether a given menu item is currently visible or not. This is called when the plugin is first loaded and every time a menu item has been pressed
- **IsMenuItemEnabled** returns whether a given menu item is currently enabled or not. This is called when the plugin is first loaded and every time a menu item has been pressed
- **OnMenuItemPressed** is called when a user selects one of the plugin's menu items. The key of the menu item (as defined by MenuLayout) is passed to the method to indicate which item has been pressed

Note that because `IsMenuItemVisible` and `IsMenuItemEnabled` are called every time a menu item has been pressed, a plugin can include logic to dynamically change the structure of its menu by changing the visibility of menu items in response to user actions. This is described in more detail in section *Writing a Plugin With a Dynamic Menu*.

Using what we've learned, it's now possible to add a menu with 1 item that displays a message box when pressed, to the example plugin. The code looks like this:

```
using System.ComponentModel.Composition;
using System.Text;
using System.Windows.Forms;
using DB.Extensibility.Contracts;

class MenuKeys
{
    public const string Root = "root";
    public const string ShowMessage = "showMessage";
}

public string MenuLayout
{
    get
    {
        StringBuilder menu = new StringBuilder();
        menu.AppendFormat(
            "*Example Plugin,{0}", MenuKeys.Root);
        menu.AppendFormat(
            ">Show Message,{0}", MenuKeys.ShowMessage);
        return menu.ToString();
    }
}

public bool IsMenuItemVisible(string key)
{
    return true;
}

namespace DB.Extensibility.Plugins
{
    [Export(typeof(IPlugin))]
    public class ExamplePlugin : PluginBase, IPlugin
    {
        class MenuKeys
        {
            public const string Root = "root";
            public const string ShowMessage = "showMessage";
        }

        public bool HasMenu
        {
            get
            {
                return true;
            }
        }
    }
}
```

```

    }
}

public string MenuLayout
{
    get
    {
        StringBuilder menu = new StringBuilder();
        menu.AppendFormat(
            "*Example Plugin,{0}", MenuKeys.Root);
        menu.AppendFormat(
            ">Show Message,{0}", MenuKeys.ShowMessage);
        return menu.ToString();
    }
}

public bool IsMenuItemVisible(string key)
{
    return true;
}

public bool IsMenuItemEnabled(string key)
{
    return true;
}

public void OnMenuItemPressed(string key)
{
    return true;
}

public void OnMenuItemPressed(string key)
{
    if (key == MenuKeys.ShowMessage)
    {
        MessageBox.Show("Menu item pressed!");
    }
}

public void Create()
{
}

public void Create()
{
}
}
}

```

- Before continuing with the example, notice the `IPlugin.Create` method. This method is the first method called after DesignBuilder has loaded a plugin. It is only called once and can be used to initialise a plugin.

If a plugin is to use the DesignBuilder API outside of the simulation hook methods (which are all passed an instance of the API), it is recommended that the `Api.Environment` object is stored as a member variable of the plugin at this point.

- Continuing on, once the plugin has been built, the only thing left to do is to copy the assembly to a location that DesignBuilder expects to find plugins.

Navigate to your local application data folder (usually, this can be found at `C:\Users\username\AppData\Local\DesignBuilder`, where “username” is your user name. The AppData directory may be hidden in which case you’ll have to enable “Show hidden files, folders, and drives” in

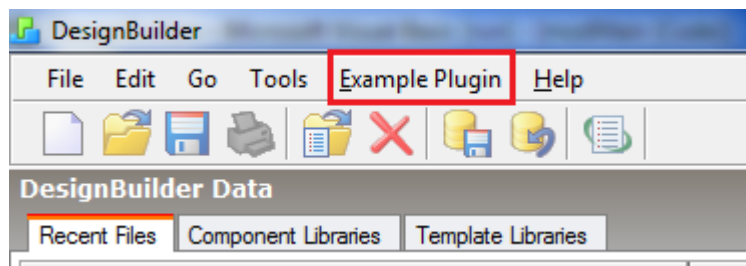
Windows Control Panel > Folder Options > View).

If there is no sub-directory called "User Plugins", create one.

Each plugin assembly should be placed into its own directory in the "DesignBuilder/User Plugins" directory (the name of the directory does not matter). Note that DB.Api.dll and DB.Extensibility.Contracts.dll must also be present in this directory as well as any external assemblies that your plugin relies upon (E.G. EPNet.dll)

DesignBuilder will load all assemblies that export a class that implements IPlugin via MEF. If you do not wish a plugin to be loaded it must be removed from the User Plugins directory (or the Export class attribute can be removed and the plugin assembly rebuilt).

If everything went according to plan, you should see the following when you start DesignBuilder:



- Pressing Example Plugin > Show Message should display a message box with the message "Menu item pressed!".

Writing a Plugin With a Dynamic Menu

Following on from the section *Writing a Plugin*, this section will show you how to exploit IPlugin.IsMenuItemVisible and IPlugin.IsMenuItemEnabled to create a menu that can change based on user interaction.

This section assumes you've already read *Writing a Plugin* and will therefore not explain the process in as much detail.

For this plugin, we are going to create a menu that can change the visibility and state of its menu items based on the user's interaction. As a base we will start with the following code:

```
[Export(typeof(IPlugin))]
public class ExamplePlugin : PluginBase, IPlugin
{
    class MenuKeys
    {
        public const string Root = "root";
    }

    public bool HasMenu
    {
        get
        {
            {
            }
        }
    }

    public string MenuLayout
    {
        get
        {
            StringBuilder menu = new StringBuilder();
            menu.AppendFormat(
                "*Example Plugin,{0}", MenuKeys.Root);
            return menu.ToString();
        }
    }

    public bool IsMenuItemVisible(string key)
    {
        {
            public const string Root = "root";
        }
    }
}
```

```

public bool HasMenu
{
    get
    {
        return true;
    }
}

public string MenuLayout
{
    get
    {
        StringBuilder menu = new StringBuilder();
        menu.AppendFormat(
            "*Example Plugin,{0}", MenuKeys.Root);
        return menu.ToString();
    }
}

public bool IsMenuItemVisible(string key)
{
    return true;
}

public bool IsMenuItemEnabled(string key)
{
    return true;
}

public void OnMenuItemPressed(string key)
{
}

public void Create()
{
}

public bool IsMenuItemEnabled(string key)
{
    return true;
}

public void OnMenuItemPressed(string key)
{
}

public void Create()
{
}
}

```

1. It's important to understand that when DesignBuilder loads a plugin for the first time, it will call `IPlugin.HasMenu` followed by `IPlugin.MenuLayout` (assuming `HasMenu` return true). This process is only carried out once because DesignBuilder needs to know the complete menu layout before it displays it's GUI. `IPlugin.IsMenuItemVisible` and `IPlugin.IsMenuItemEnabled`, however, are called throughout the lifetime of the plugin.

This is important because it means that `IPlugin.MenuLayout` must return the complete menu structure to DesignBuilder and only through toggling visibility can the structure be changed.

With that in mind, add the following code to the plugin:

```
[Export(typeof(IPlugin))]
public class ExamplePlugin : PluginBase, IPlugin
{
    class MenuKeys
    {
        public const string Root = "root";
        public const string State = "state";
        public const string Visibility = "visibility";
        public const string EnableAll = "enableAll";
        public const string DisableAll = "disableAll";
        public const string VisibleAll = "visibleAll";
        public const string InvisibleAll = "invisibleAll";
    }

    ...

    public string MenuLayout
    {
        get
        {
            StringBuilder menu = new StringBuilder();
            menu.AppendFormat(
                "*Example Plugin,{0}", MenuKeys.Root);
            menu.AppendFormat(
                ">State,{0}", MenuKeys.State);
            menu.AppendFormat(
                ">>Enable All,{0}", MenuKeys.EnableAll);
            menu.AppendFormat(
                ">>Disable All,{0}", MenuKeys.DisableAll);
            menu.AppendFormat(
                ">Visibility,{0}", MenuKeys.Visibility);
            menu.AppendFormat(
                ">>Make All Visible,{0}", MenuKeys.VisibleAll);
            menu.AppendFormat(
                ">>Make All Invisible,{0}", MenuKeys.InvisibleAll);
            return menu.ToString();
        }
    }

    ...
}
```

- Now the menu structure is defined the plugin will need some way of keeping track of the state of each menu item. I.E. Whether or not a menu item is visible or enabled. This can be done however you like, but in this example we shall create a simple class and a dictionary:

```

using System.Collections.Generic;

...

[Export(typeof(IPlugin))]
public class ExamplePlugin : PluginBase, IPlugin
{
    ...

    class MenuItem
    {
        public bool IsEnabled { get; set; }
        public bool IsVisible { get; set; }

        public MenuItem(
            bool enabled = true,
            bool visible = true)
        {
            IsEnabled = enabled;
            IsVisible = visible;
        }
    }

    private readonly Dictionary<string, MenuItem> mMenuItems =
        new Dictionary<string, MenuItem>();

    ...

    public void Create()
    {
        mMenuItems.Add(MenuKeys.Root, new MenuItem());
        mMenuItems.Add(MenuKeys.State, new MenuItem());
        mMenuItems.Add(MenuKeys.Visibility, new MenuItem());
        mMenuItems.Add(MenuKeys.EnableAll, new MenuItem());
        mMenuItems.Add(MenuKeys.DisableAll, new MenuItem());
        mMenuItems.Add(MenuKeys.VisibleAll, new MenuItem());
        mMenuItems.Add(MenuKeys.InvisibleAll, new MenuItem());
    }
}

```

Notice that `IPlugin.Create` is used to initialise the plugin. You could do this in the constructor if you prefer.

3. `IPlugin.IsMenuItemVisible` and `IPlugin.IsMenuItemEnabled` can now simply return the `IsVisible` or `IsEnabled` properties of the specific menu item like so:

```
[Export(typeof(IPlugin))]
public class ExamplePlugin : PluginBase, IPlugin
{
    ...

    public bool IsMenuItemVisible(string key)
    {
        return mMenuItems[key].IsVisible;
    }

    public bool IsMenuItemEnabled(string key)
    {
        return mMenuItems[key].IsEnabled;
    }

    ...
}
```

The menu structure is defined, the plugin has a means of keeping track of each menu item's state, and it can now report the state back to DesignBuilder. All that remains is to actually change the state of a menu item depending on the actions of the user.

Since each menu item will have a different action when pressed by the user, we will add a property to the nested MenuItem class to represent this:

```
class MenuItem
{
    public Action Action { get; set; }
    public bool IsEnabled { get; set; }
    public bool IsVisible { get; set; }

    public MenuItem(
        Action action = null,
        bool enabled = true,
        bool visible = true)
    {
        Action = action ?? delegate{};
        IsEnabled = enabled;
        IsVisible = visible;
    }
}
```

And then initialise each menu item with an appropriate action:

```
public class ExamplePlugin : PluginBase, IPlugin
{
    ...

    public void Create()
    {
        mMenuItems.Add(MenuKeys.Root,
            new MenuItem());
        mMenuItems.Add(MenuKeys.State,
            new MenuItem());
        mMenuItems.Add(MenuKeys.Visibility,
            new MenuItem());
        mMenuItems.Add(MenuKeys.EnableAll,
            new MenuItem(OnEnableAll));
        mMenuItems.Add(MenuKeys.DisableAll,
            new MenuItem(OnDisableAll));
        mMenuItems.Add(MenuKeys.VisibleAll,
            new MenuItem(OnVisibleAll));
        mMenuItems.Add(MenuKeys.InvisibleAll,
            new MenuItem(OnInvisibleAll));
    }
}
```

```

private void OnEnableAll()
{
    mMenuItems[MenuKeys.DisableAll].IsEnabled = true;
    mMenuItems[MenuKeys.Visibility].IsEnabled = true;
}

private void OnDisableAll()
{
    // don't disable EnableAll
    mMenuItems[MenuKeys.DisableAll].IsEnabled = false;
    mMenuItems[MenuKeys.Visibility].IsEnabled = false;
}

private void OnVisibleAll()
{
    mMenuItems.Add(MenuKeys.Root,
        new MenuItem());
    mMenuItems.Add(MenuKeys.State,
        new MenuItem());
    mMenuItems.Add(MenuKeys.Visibility,
        new MenuItem());
    mMenuItems.Add(MenuKeys.EnableAll,
        new MenuItem(OnEnableAll));
    mMenuItems.Add(MenuKeys.DisableAll,
        new MenuItem(OnDisableAll));
    mMenuItems.Add(MenuKeys.VisibleAll,
        new MenuItem(OnVisibleAll));
    mMenuItems.Add(MenuKeys.InvisibleAll,
        new MenuItem(OnInvisibleAll));
}

private void OnEnableAll()
{
    mMenuItems[MenuKeys.DisableAll].IsEnabled = true;
    mMenuItems[MenuKeys.Visibility].IsEnabled = true;
}

private void OnDisableAll()
{
    // don't disable EnableAll
    mMenuItems[MenuKeys.DisableAll].IsEnabled = false;
    mMenuItems[MenuKeys.Visibility].IsEnabled = false;
}

private void OnVisibleAll()
{
    mMenuItems[MenuKeys.InvisibleAll].IsVisible = true;
    mMenuItems[MenuKeys.State].IsVisible = true;
}

private void OnInvisibleAll()
{
    // don't make VisibleAll invisible
    mMenuItems[MenuKeys.InvisibleAll].IsVisible = false;
    mMenuItems[MenuKeys.State].IsVisible = false;
}
}

```

Note that changing the visibility or enabled state of a parent menu item also affects the state of its children. E.G. OnDisableAll changes MenuKeys.Visibility's enabled state to false, which results in MenuKeys.VisibleAll and MenuKeys.InvisibleAll also being disabled.

- Each relevant menu item now has an appropriate action or a default action to do nothing. The final piece in the puzzle is to forward IPlugin.OnMenuItemPressed to the correct menu item's action as follows:

```
[Export(typeof(IPlugin))]
public class ExamplePlugin : PluginBase, IPlugin
{
    ...

    public void OnMenuItemPressed(string key)
    {
        mMenuItems[key].Action();
    }

    ...
}
```

5. The plugin is now complete and can be copied to the User Plugin directory as described in section *Writing a Plugin* item 7. Upon starting DesignBuilder the plugin should be loaded and you can see how pressing Example Plugin > Visibility > Make All Invisible changes the structure of the menu (from the user's point-of-view).

Plugin API Versions

When new plugin hookpoints are introduced a new plugin interface is created, which inherits the previous plugin interface. This ensures that plugins that support an older interface will still work without the plugin author having to change their code.

DesignBuilder supports two plugin interfaces, IPlugin and IPlugin2. In addition to supporting all IPlugin's hook points, IPlugin2 adds three new hook points – ScreenChanged, ModelLoaded, and ModelUnloaded. For your plugin to take advantage of the latest API you must define your plugin class accordingly:

```
[Export(typeof(IPlugin2))]
public class ExamplePlugin : PluginBase2, IPlugin2
{
}
```

IPlugin supported hook points

This is a list of the supported hook points in the IPlugin interface:

```
void BeforeEnergyIdfGeneration();
void BeforeEnergySimulation();
void AfterEnergySimulation();
void BeforeHeatingIdfGeneration();
void BeforeHeatingSimulation();
void AfterHeatingSimulation();
void BeforeCoolingIdfGeneration();
void BeforeCoolingSimulation();
void AfterCoolingSimulation();
void BeforeDaylightSimulation();
void AfterDaylightSimulation();
void BeforeCfdSimulation();
void AfterCfdSimulation();
void BeforeOptimisationStudy();
void AfterOptimisationStudy();
void OnDesignVariableChanged(int variableId, string value);
void BeforeCostAndCarbon();
void AfterCostAndCarbon();
void AtCommandLine(string arg);
```

IPlugin2 supported hook points

This is a list of the supported hook points in the IPlugin2 interface (in addition to the above IPlugin hook points):

```
void ModelLoaded();  
void ModelUnloaded();  
void ScreenChanged(ScreenCode screenCode);
```

Working with the DesignBuilder API

This section provides background information on how to work with the DesignBuilder Building Model using the API. The examples provided assume that you are developing a plugin using C# code but the same principles apply for C# and Python scripts.

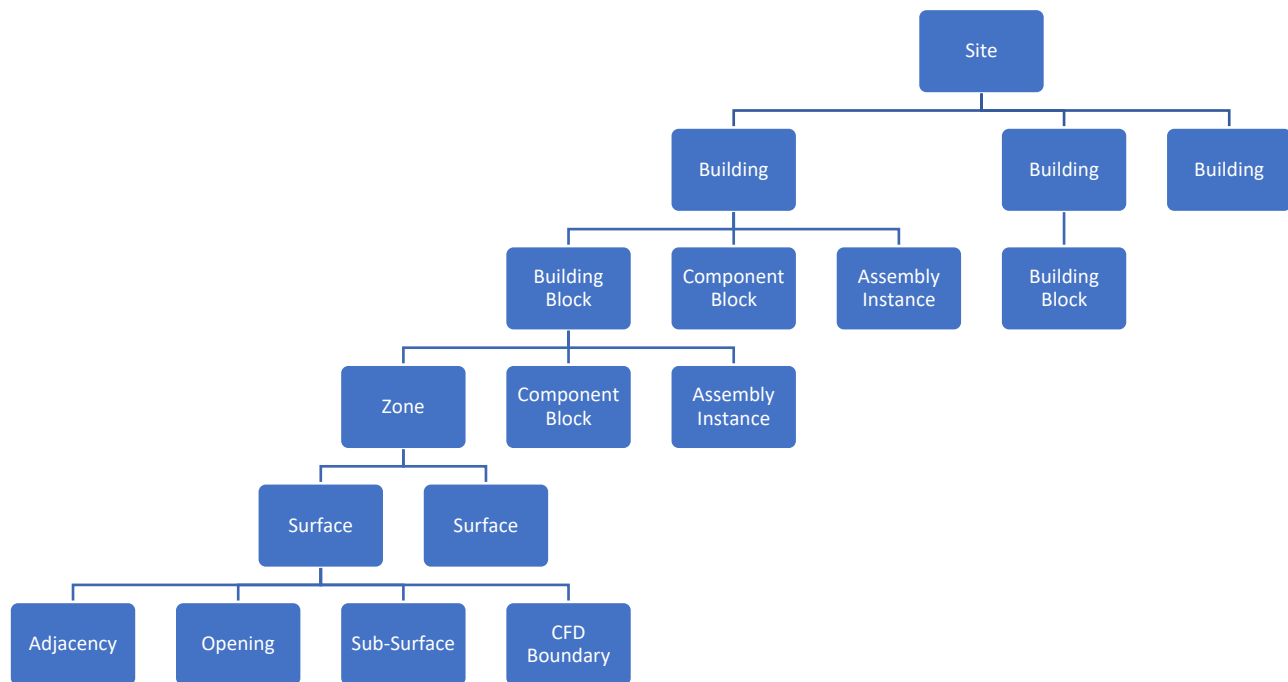
DesignBuilder Building Model

The DesignBuilder building model consists of a hierarchy of objects that collectively represent a building or collection of buildings within a site. The hierarchy is essentially a tree like structure that starts with a single **Site** object at the top.

The **Site** object acts as a container for a single **Building** object or list of **Building** objects. A **Building** object at its simplest is composed of a single **Building Block** object or a list of **Building Block** objects but may also contain **Component Blocks** and **Assembly Instances**. A **Building Block** in turn at its simplest comprises a **Zone** or number of **Zones** but may also contain **Component Blocks** and **Assembly Instances**. **Zones** are composed of a number of **Surfaces** where each surface represents one of the bounding elements of the zone (walls, floors, ceilings/roofs). Each surface has at least one **Adjacency** and may have a number of **Openings**, **Sub-Surfaces** and/or **CFD Boundaries**. An **Adjacency** is a segment of the surface, the geometry of which is essentially a polygon or list of polygons that represent the geometric portion of the surface that is adjacent to the outside of the building or to another zone in the building, an adjacency also contains additional information such as construction, etc. **Opening** objects represent any opening within the surface, including windows, doors and holes. **Sub-surfaces** are similar to Openings but are used to represent segments of a surface that have a different construction to the main area of the surface. **CFD Boundaries** are also similar to openings but are used to model various CFD boundary conditions including temperature and heat flux patches as well as ventilation boundaries. **Adjacencies**, **Openings**, **Sub-Surfaces** and **CFD Boundaries** are the lowest level objects in the DesignBuilder model hierarchy.

In a similar fashion to **Adjacencies**, the geometry of **Openings**, **Sub-Surfaces** and **CFD Boundaries** are essentially simple polygons and the objects also contain additional information concerning construction, etc.

Each level in the hierarchy is known as a **Level of Decomposition (Site/Building/Building Block/Zone/Surface/Opening)**. At each **Level of Decomposition**, a number of different types of model object may be present (**Building Blocks**, **Component Blocks** and **Assembly Instances** at **Building level** and **Zones**, **Component Blocks** and **Assembly Instances** at **Building Block level**). These objects are known as **Decomposition Level Objects**.



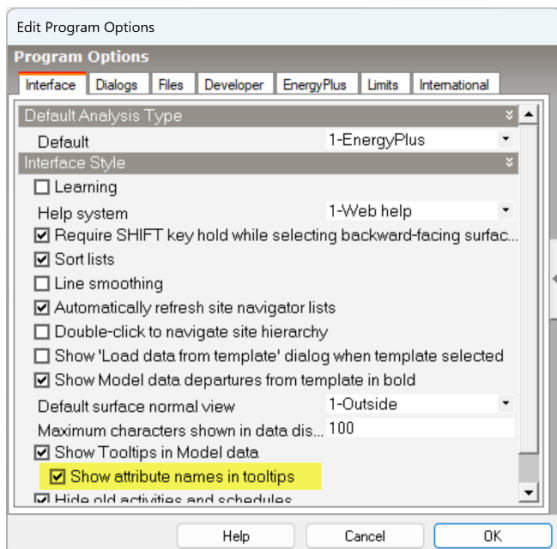
The various **Decomposition Level Objects** (**Site**, **Building**, **Building Block**, etc.) have corresponding API classes associated with them which provide various levels of functionality associated with the objects including access to attribute data. The API classes generally have the same names as the associated model objects.

Decomposition Level Object Data and Attributes

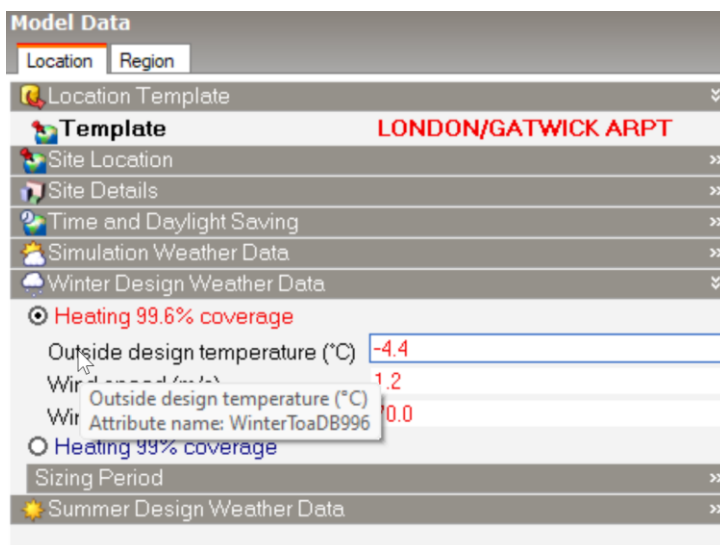
All **Decomposition Level Objects** (**Site**, **Building**, **Building Block**, etc.) store data via **Attribute** objects which are simple pairs of string where one string is the name of the attribute and the other string is the value of the attribute. DesignBuilder employs an inheritance mechanism which allows object data stored in attributes to be automatically inherited from one **Level of Decomposition** to all levels below that level. If any item of data is changed from the inherited value at any **Level of Decomposition**, all objects below that level will automatically take on or inherit the value at that level. Attributes set at a particular level are known as **hard** attributes.

So, for example, if a design heating setpoint temperature of say 19.0 C is set at the **Building** level of decomposition, this setpoint will automatically be applied to all building blocks and building block zones within the building. However, if all zones in one particular building block require a setpoint temperature of 20.0 C, the setting can be made at the building block level and then all zones within that building block will automatically inherit the setpoint temperature of 20.0 C.

When developing application plugins, the application will normally require access to items of data belonging to various model objects which have been set via the DesignBuilder UI. To access this data, knowledge of the associated attribute name will be required. To facilitate this, a feature is built into the DesignBuilder UI to enable settings tooltips to include the name of the associated attribute. To switch this feature on, a setting is provided on the Program Options Dialog which can be accessed from the Tools menu:



So, for example, to access the name of the attribute behind the site level 'Winter outside design temperature' you would navigate to the site level and move the cursor over the 'Outside design temperature' setting under the 'Winter Design Weather Data' header and wait for the tooltip to appear:



The attribute name **WinterToaDB996** appears on the tooltip. This name can then be used to extract the value of the setting from the site level object attribute:

```
outsideAirTemperature = site.GetAttributeAsDouble("WinterToaDB996");
```

Notice that the method used to obtain the value of the site level attribute returns the value as a double. There are a number of such methods associated with the various API model objects that can be used to access attributes. These methods are described more fully in the next sections.

The Site Object, Model Status and the API Environment Class

The highest level object in the model hierarchy is the **Site** object. The **Site** object enables access to all other objects in the model hierarchy as well as access to all of the tables that contain model data such as constructions, materials, glazing, etc. The **Site** object is accessed via the **Environment** object.

During start-up, DesignBuilder instantiates a single **DB.Api.Environment** object. This singleton object is used by all active scripts and plugins, and consequently developers should not create their own objects of this class

but instead rely on the instance incorporated within the plugin, the **ApiEnvironment** object. As well as providing access to the model **Site** object, the **ApiEnvironment** object may also be used to access the model status at the time of running the plugin. The model status includes various model states such as the current level of decomposition, current decomposition level object type, current building index, current zone index, etc.

In the example below the menu item handler **OnMenuItemPressed** uses the **ApiEnvironment** object to first make sure that the model contains a finite number of buildings and then checks that the current decomposition level is **Building**, it then obtains the current building index and uses it to access the current building (a site may contain several buildings). A further check is then carried out to make sure that the current building contains a finite number of building blocks (and consequently zones):

```
public override void OnMenuItemPressed(string key)
{
    Site modelSite = ApiEnvironment.Site;

    if ((IsModelLoaded) && (ApiEnvironment.Site.Buildings.Count > 0) && (ApiEnvironment.DecompositionLevel == DecompositionLevel.Building) &&
        (ApiEnvironment.Site.Buildings[ApiEnvironment.CurrentBuildingIndex].BuildingBlocks.Count > 0))
    {
        var heatLossDlg = new MainForm();

        // initialisation checks for valid U-values
        if(heatLossDlg.Initialise(modelSite, modelSite.Buildings[ApiEnvironment.CurrentBuildingIndex]))
            heatLossDlg.ShowDialog();
    }
    else
    {
        MessageBox.Show("Model must be at building level and the building must contain zones for heat loss calculation");
    }
}
```

For further details of the plugin framework, please refer to [Extending DesignBuilder with Plugins](#).

DesignBuilder Tables

DesignBuilder makes extensive use of Table objects for a range of purposes. Most tables come into the following categories:

1. To hold databases of components and templates.
2. To hold results of calculations.
3. To define the layout of dialog.

The various types of Table are stored and accessed as follows:

1. All *model* component and template database tables are stored at site level and accessed from the site object. For example to access the model **Constructions** table use code like:


```
ConstructionTable = ApiEnvironment.Site.GetTable("Constructions")
```
2. All *library* components and templates are stored in the **DB.Api.ApplicationComponents** and **DB.Api.ApplicationTemplates** objects respectively. Library components and templates are those that are defined from the opening screen. When a new model is created the library components and templates that may be associated with models are copied to the site level of the new model. Some databases are associated with the Application (e.g. **ProgramOptionTemplates**) and so are not copied to the model. These are listed at [Appendix 1 List of Available Tables](#) summary with "Y" in the [Application](#) column and in the [TableOfTables](#) with "True" in the **IsApplicationData** field.
3. All *dialog layout* tables are stored in and accessed from the **ApplicationTemplates** object since they are not model-dependent.
4. Predefined option lists. These are a special case of component database tables but are non-editable. These are listed at [Appendix 1 List of Available Tables](#) summary with "N" in the [Editable](#) column and in the [TableOfTables](#) with "False" in the **IsEditable** field.

5. Tables containing *results of simulations* are stored with the appropriate model object. For example building level results are stored with and accessed from the building model object. Likewise results for blocks, zones, surfaces and openings are stored with the corresponding object in the model. See below in the [Results tables](#) section for more information on these tables.

Component and Template database tables

All DesignBuilder tables consist of a number of records and each record includes a number of fields. Records and fields can be thought of as rows and columns respectively.

In component and template database tables, the first field (index 0) is normally the Id of the record which can be used to access a specific record in the database and subsequent fields contain the rest of the data.

The **Table** object has a **Records** method which returns a collection of records. This method can then be used to access specific records using the record Id.

Category Id field references a value in the first record of the table. The Ids in this first record point to a category string in an externalised text file.

#4	#5	#6	#7	#8	#9	#10	#11
#Id	#Name	#CategoryId	#U-Value	#SRO	#SRI	#HCO	#HCI
#2055	#Roof, Metal Building, R-0 (0.0), U-1.280 (7.258)	#5	#7.1304840790910	#.04	#.1	#19.8700008392334	#4.45959997177124
#2056	#Roof, Metal Building, R-6 (1.1), U-0.167 (0.947)	#5	#.944422970062801	#.04	#.1	#19.8700008392334	#4.45959997177124
#2057	#Roof, Metal Building, R-10 (1.8), U-0.097 (0.551)	#5	#.550000000000000	#.04	#.1	#19.8700008392334	#4.45959997177124
#2058	#Roof, Metal Building, R-11 (1.9), U-0.092 (0.522)	#5	#.552101987066949	#.04	#.1	#19.8700008392334	#4.45959997177124
#2059	#Roof, Metal Building, R-13 (2.3), U-0.083 (0.471)	#5	#.47030287403147	#.04	#.1	#19.8700008392334	#4.45959997177124
#2060	#Roof, Metal Building, R-16 (2.8), U-0.072 (0.409)	#5	#.408626977190989	#.04	#.1	#19.8700008392334	#4.45959997177124
#2061	#Roof, Metal Building, R-19 (3.3), U-0.065 (0.369)	#5	#.368685893297776	#.04	#.1	#19.8700008392334	#4.45959997177124
#2062	#Roof, Metal Building, R-10+10 (1.8+1.8), U-0.063 (0.357)	#5	#.356460586108767	#.04	#.1	#19.8700008392334	#4.45959997177124
#2063	#Roof, Metal Building, R-10+11 (1.8+1.9), U-0.061 (0.346)	#5	#.345574595135761	#.04	#.1	#19.8700008392334	#4.45959997177124
#2064	#Roof, Metal Building, R-11+11 (1.9+1.9), U-0.060 (0.340)	#5	#.339570801305902	#.04	#.1	#19.8700008392334	#4.45959997177124
#2065	#Roof, Metal Building, R-10+13 (1.8+2.3), U-0.058 (0.329)	#5	#.328669668168965	#.04	#.1	#19.8700008392334	#4.45959997177124
#2066	#Roof, Metal Building, R-11+13 (1.9+2.3), U-0.057 (0.323)	#5	#.32274906265803	#.04	#.1	#19.8700008392334	#4.45959997177124
#2067	#Roof, Metal Building, R-13+13 (2.3+2.3), U-0.055 (0.312)	#5	#.311751345243025	#.04	#.1	#19.8700008392334	#4.45959997177124
#2068	#Roof, Metal Building, R-10+19 (1.8+3.3), U-0.052 (0.295)	#5	#.294660929439133	#.04	#.1	#19.8700008392334	#4.45959997177124
#2069	#Roof, Metal Building, R-11+19 (1.9+3.3), U-0.051 (0.289)	#5	#.288725395354349	#.04	#.1	#19.8700008392334	#4.45959997177124
#2070	#Roof, Metal Building, R-13+19 (2.3+3.3), U-0.049 (0.278)	#5	#.277723167586953	#.04	#.1	#19.8700008392334	#4.45959997177124
#2071	#Roof, Metal Building, R-16+19 (2.8+3.3), U-0.047 (0.266)	#5	#.265710204659685	#.04	#.1	#19.8700008392334	#4.45959997177124
#2072	#Roof, Metal Building, R-19+19 (3.3+3.3), U-0.046 (0.261)	#5	#.26071556910915	#.04	#.1	#19.8700008392334	#4.45959997177124
#2073	#Roof 38 - Metal building Roof - Metal roof panel, R-11+19	#5	#.190639574717228	#.04	#.1	#19.8700008392334	#4.45959997177124
#2074	#Roof, Metal Building, R-10 (1.8), U-0.153 (0.868)	#5	#.866403687577495	#.04	#.1	#19.8700008392334	#4.45959997177124
#2075	#Roof, Metal Building, R-11 (1.9), U-0.139 (0.788)	#5	#.785668748204715	#.04	#.1	#19.8700008392334	#4.45959997177124
#2076	#Roof, Metal Building, R-13 (2.3), U-0.130 (0.737)	#5	#.735917755168495	#.04	#.1	#19.8700008392334	#4.45959997177124
#2077	#Roof, Metal Building, R-16 (2.8), U-0.106 (0.606)	#5	#.604523146081859	#.04	#.1	#19.8700008392334	#4.45959997177124
#2078	#Roof, Metal Building, R-19 (3.3), U-0.098 (0.550)	#5	#.548956420761957	#.04	#.1	#19.8700008392334	#4.45959997177124
#2079	#Roof 44 - Metal building Roof - Metal roof panel, R-11+19	#5	#.222738685618652	#.04	#.1	#19.8700008392334	#4.45959997177124
#2080	#Roof, Metal Building, R-19+10 (3.3+1.8), U-0.041 (0.232)	#5	#.231892884825321	#.04	#.1	#19.8700008392334	#4.45959997177124
#2081	#Roof, Ins Entirely above Deck, R-0 (0.0), U-1.282 (7.280)	#5	#7.13160713152083	#.04	#.1	#19.8700008392334	#4.45959997177124
#2082	#Roof, Ins Entirely above Deck, R-1 (0.2), U-0.562 (3.191)	#5	#3.17237888401835	#.04	#.1	#19.8700008392334	#4.45959997177124
#2083	#Roof, Ins Entirely above Deck, R-2 (0.4), U-0.360 (2.044)	#5	#2.02840321434469	#.04	#.1	#19.8700008392334	#4.45959997177124
#2084	#Roof, Ins Entirely above Deck, R-3 (0.5), U-0.265 (1.505)	#5	#1.4970090378852	#.04	#.1	#19.8700008392334	#4.45959997177124
#2152	#Roof, Ins Entirely above Deck, R-4 (0.7), U-0.218 (1.240)	#5	#1.23507817232037	#.04	#.1	#19.8700008392334	#4.45959997177124

The above screenshot shows an extract from the **Constructions** table source file **Constructions.dat**.

You can find a copy of the dat files in the DesignBuilder\Data folder which can be accessed from within DesignBuilder using the "File > Folders > Library data" folder menu command.

The first 2 lines contain the category index lists and the field names respectively. The actual data starts on the third row which is record index 0.

Note the first Id field is used to access records. As an example of how this is used, the following code fragment shows how the construction Id for an internal partition adjacency is first obtained from the adjacency **InternalWallConstr** attribute and then used to obtain the associated record from the Constructions table using the **GetRecordFromHandle** method of the **Records** collection class:

```

case SurfaceType.InternalPartition:
    constructionId = Convert.ToInt32(adjacency.GetAttribute("InternalWallConstr"));
    break;
default: break;
}

if (!(constructionId == -1)) record = constructionTable.Records.GetRecordFromHandle(constructionId);
return (record);

```

A full list of available component and template tables can be found in [Appendix 1 List of available tables](#)

Categories

Each component and template database table includes a list of category Ids each of which points to externalised text in the LocalisedCategory_1.txt category text file. The number 1 denotes the English text and equivalent files exist for other languages as well.

Each data record in the database includes a category field called CategoryId (the third field) that contains an index into the category text file.

Results tables

Results tables are accessed from the relevant model object using interval and 2-letter calculation codes as illustrated below.

The list of interval codes is:

- **Summary** – summary results
- **Annual** – annual results
- **Monthly** – monthly results
- **Daily** – daily results
- **Hourly** – hourly results
- **TimeSteply** – timestep results

The list of 2-letter calculation codes is:

- **SS** – simulation
- **HG** – cooling design
- **HL** – heating design

So, for example, to access the **summary cooling design** results table for a building, use code like:

```
BuildingSummaryTable = building.GetTable("HGSummary")
```

Or, to obtain a table with **annual simulation** zone level results:

```
ZoneAnnualTable = zone.GetTable("SSAnnual")
```

The easiest way to learn how to work with tables is to study example script and plugin code such as that provided in the [DesignBuilder GitHub repository](#).

Navigating Model Objects via the API

The API classes associated with the various **Decomposition Level** objects have methods to obtain collections of objects that can be housed at each particular level. For example the **Building** class has a **BuildingBlocks** method that returns a collection of all building blocks within the building and the **BuildingBlock** class has a **Zones** method that returns a collection of building zones. These methods can be used to iterate through the model hierarchy:

```

public void PopulateZoneDataGridView()
{
    double insideAirTemperature, infiltration, totalUAValue, outsideAirTempertaure;
    double volume, qf, qv, qbl, qt, dt, blpsi;
    string name;

    ZoneDataGridView.Rows.Clear();

    outsideAirTempertaure = site.GetAttributeAsDouble("WinterToaDB996");

    foreach (BuildingBlock buildingBlock in building.BuildingBlocks)
    {
        foreach (Zone zone in buildingBlock.Zones)
        {
            name = zone.GetAttribute("Title");
            insideAirTemperature = zone.GetAttributeAsDouble("HeatingDesignSetPointTemperature");
            volume = zone.Volume;
            infiltration = zone.GetAttributeAsDouble("InfiltrationValue");

            totalUAValue = 0.0;
        }
    }
}

```

Zone Surface Types and Holes

Each zone surface has an associated type which refers to the zone element type by orientation: Wall, Flat Roof (Ceiling), Floor, Pitched Roof, etc. The API [Surface](#) class incorporates a [Type](#) which allows the type of the surface to be determined. Surface types are listed within the API enumerated type [SurfaceType](#). Surface types are important when determining construction information due to the way that the attribute inheritance system works. A particular adjacency or surface construction may not be explicitly stored at the adjacency or surface level but may be inherited from a higher level. Therefore when extracting construction data from an adjacency or surface level, the surface type must be used in order to determine which attribute to use. This is demonstrated in the example code below:

```

switch (surface.Type)
{
    case SurfaceType.Floor:
        if (adjacency.IsExternal)
        {
            switch (adjacency.AdjacencyCondition)
            {
                case AdjacencyCondition.AdjacentToGround:
                    constructionId = Convert.ToInt32(adjacency.GetAttribute("CombinedGroundFloorConstr"));
                    break;

                case AdjacencyCondition.Adiabatic:
                    constructionId = Convert.ToInt32(adjacency.GetAttribute("CombinedInternalFloorConstr"));
                    break;

                default:
                    constructionId = Convert.ToInt32(adjacency.GetAttribute("CombinedExternalFloorConstr"));
                    break;
            }
        }
        else
        {
            constructionId = Convert.ToInt32(adjacency.GetAttribute("CombinedInternalFloorConstr"));
        }

        break;
}

```

In this code fragment, the surface type is used to determine the element type that the adjacency belongs to (floor, wall, etc.), the adjacency is then checked to see if it's an external or internal adjacency. If the surface type is a floor and the adjacency is external, the [Adjacency](#) method [AdjacencyCondition](#) is then used to determine if the external adjacency is adiabatic, adjacent to the ground or internal in order to finally select the type of construction attribute from which to obtain the construction record Id.

When iterating through zone surface lists, special attention needs to be paid to surfaces representing holes. Zones containing voids, courtyards or nested zones will incorporate holes in ceiling and floor surfaces. These holes are represented as surfaces themselves within the zone surface list and would normally need to be discounted during the processing of zone surface lists.

A temporary surface list can be created for a zone which excludes all surface holes:

```
public void BuildSurfaceList(Zone zone)
{
    surfaceList.Clear();

    foreach (Surface surface in zone.Surfaces) if (!(surface.Type == SurfaceType.Hole))
        surfaceList.Add(surface);
}
```

Extracting Geometry

For certain applications, it may be desirable to obtain the underlying Cartesian geometry (coordinate information) that represents zone surfaces and surface elements (adjacencies and openings). The coordinate information may be obtained for surfaces and surface elements by means of **Polygon** objects. A **Polygon** object in the API is a collection of 3D points or vertices. A **Surface** object has a single **Polygon** associated with it which can be obtained using the **SurfacePolygon** method. So, for example, to obtain a list of polygons representing the geometry of a zone, the following code could be used:

```
public List<Polygon> GetZoneGeometry(Zone zone)
{
    List<Polygon> polygonList;

    polygonList = new List<Polygon>();

    BuildSurfaceList(zone);

    for (int i = 0; (i < surfaceList.Count()); i++) polygonList.Add(surfaceList[i].SurfacePolygon);

    return (polygonList);
}
```

The vertex information can then be obtained for each polygon by looping through the polygon vertices:

```

public void ProcessZoneGeometry(Zone zone)
{
    List<Polygon> polygonList;

    polygonList = GetZoneGeometry(zone);

    for(int i=0; (i < polygonList.Count); i++)
    {
        for(int j=0; (j<polygonList[i].Vertices.Count); j++)
        {
            // Obtain X, Y, Z coordinates for each vertex using polygonList[i].Vertices[j].X,
            // polygonList[i].Vertices[j].Y, polygonList[i].Vertices[j].Z
            // ....
            // ....
            //....
            //....
        }
    }
}

```

Setting Startup Location

When a model is saved, DesignBuilder stores the location in the model that the user is currently navigated to. This allows it to open at the same location when the file is reloaded. The location is stored in the "SiteStartupOptions" site-level attribute.

It can at times be useful to control this location. For example, when loading models from the command line users may wish to open at a specific building in the model. This can be controlled by writing code to update the "SiteStartupOptions" attribute.

The attribute is a semi-colon delimited string that would look as below if the user was navigated to the first building when the model was last saved:

"1;0;0;-1;-1;-1;-1;-1;-1;-1;-1;-1;-1;-1;-1"

The items in the string in order are:

Meaning	Index in attribute	Comments		
Level	0	DecompositionLevel		
ObjectType	1	Name	Value in attribute	Explanation
		Hierachy	0	object is one of site, building, block, zone etc in main building hierarchy
		ComponentBlock	1	a component block
		AssemblyInstance	2	an assembly instance
		Plane	3	A shading plane loaded from a gbXML model
BuildingIndex	2			

BlockIndex	3	
ComponentBlockIndex	4	
AssemblyInstanceIndex	5	
PlaneIndex	6	
ZoneIndex	7	
SurfaceIndex	8	
OpeningIndex	9	
HVACLoopIndex	10	
HVACSubLoopIndex	11	
HVACComponentIndex	12	
HVACSubComponentIndex	13	
HVACZoneGroupIndex	14	
HVACObjectType	15	HVACObjectType

All indices are zero-based.

API Reference

The [DesignBuilder API Help Guide](#) is the main source of help on all of the API classes, collections, enumerations etc.

Example Scripts and Plugin Code Repository

DesignBuilder provides a [GitHub repository](#) of example scripts and plugin code which you can use to learn how to how about creating your own scripts and plugins.

Appendix 1 List of Available Tables

The table below lists all of the tables included in DesignBuilder models and the library. It summarises the detailed information included in the full [TableOfTables table](#).

The meaning of the data in the various columns is explained below the table.

Table Name	Editable	Application	Template	Model Data	HVAC	Comments
Constructions	Y	N	N	N	N	Constructions database
Glazing	Y	N	N	N	N	Glazing database
Materials	Y	N	N	N	N	Materials database
Panes	Y	N	N	N	N	Panes database
Textures	Y	N	N	N	N	Texture database
WindowGas	Y	N	N	N	N	Window gas database
LocalShading	Y	N	N	N	N	Local shading database
InternalBlinds	Y	N	N	N	N	Internal blinds database
Vents	Y	N	N	N	N	Vents database
Holidays2	Y	N	N	N	N	Holidays database
MetabolicRates	Y	N	N	N	N	Metabolic rates database
HolidaySchedule	Y	Y	Y	N	N	
Schedules	Y	N	N	N	N	Schedules database
Profiles	Y	N	N	N	N	Profiles database (used in 7/12 Schedules)

ActivityTemplates	Y	N	Y	N	N	Activity templates database
ConstructionTemplates	Y	N	Y	N	N	Construction templates database
GlazingTemplates	Y	N	Y	N	N	Glazing templates database
FacadeTemplates	Y	N	Y	N	N	Façade templates database
LightingTemplates	Y	N	Y	N	N	Lighting templates database
HVACTemplates	Y	N	Y	N	N	Simple HVAC templates database
DHWTemplates	Y	N	Y	N	N	DHW templates database
ProgramOptionTemplates	Y	Y	Y	N	N	Program options database. DesignBuilder uses Id = 2.
MenuTools	Y	Y	Y	N	N	List of menu options - for internal use only
LocationTemplates	Y	N	Y	N	N	Location templates database
UKDDData	N	Y	Y	N	N	UK degree day data - for internal use only
Countries	N	Y	Y	N	N	Countries database - for internal use only.
RegionsRaw	N	Y	Y	N	N	For internal use only
HourlyWeather	Y	N	N	N	N	Hourly weather database
WBANWMOLocations	Y	Y	Y	N	N	For internal use only
TimeZones	Y	Y	Y	N	N	Time zones database – for internal use only
CpData	Y	N	Y	N	N	Wind pressure coefficient database
Cracks	Y	N	Y	N	N	Cracks data used for Calculated nat vent - for internal use only
Regions	Y	N	Y	N	N	Regions database
EnergyCodes	Y	N	Y	N	N	Energy codes database
Sectors	Y	N	Y	N	N	Building usage sectors database
ECMeasures	Y	Y	Y	N	N	Not used
LoadDataFromTemplate	Y	Y	Y	N	N	Data for the Load data from template dialog
NewBuilding	Y	Y	Y	N	N	New building dialog
NewSite	Y	Y	Y	N	N	New site (model) dialog
Messages	Y	Y	Y	N	N	List of messages - for internal use only
HGDisplayOptions	Y	N	Y	N	N	Cooling design display options panel
HLDisplayOptions	Y	N	Y	N	N	Heating design display options panel
SSDisplayOptions	Y	N	Y	N	N	Simulation display options panel
CheckDisplayOptions	Y	N	Y	N	N	SBEM/DSM display options panel
UpgradeAdvice	Y	Y	Y	N	N	For internal use only
Units	Y	Y	Y	N	N	Units database
ImportDXF	Y	Y	Y	N	N	Import floor plans dialog
Import3D	Y	Y	Y	Y	N	Import 3D models dialog
EquipmentList	Y	Y	Y	N	N	Detailed equipment gains
DataOptionTemplates	Y	Y	Y	Y	N	Model options dialog
DrawingTools	Y	Y	Y	Y	N	
HGCalcOptions	Y	Y	Y	Y	N	Cooling design calculation options
HLCalcOptions	Y	Y	Y	Y	N	Heating design calculation options
SSCalcOptions	Y	Y	Y	Y	N	Simulation design calculation options

EnergyRatingCalcOptions	Y	Y	Y	Y	N	SBEM/DSM design calculation options
CFDCalcOptions	Y	Y	Y	Y	N	CFD design calculation options
ReportOptions	Y	Y	Y	Y	N	Cost and Carbon design calculation options
MovieCalcOptions	Y	Y	Y	Y	N	Movie generation dialog
ThreeDDisplayOptions	Y	Y	Y	Y	N	Visualisation display options panel
CFDDisplayOptions	Y	Y	Y	Y	N	CFD display options panel
tHEFFd	N	Y	Y	N	N	SBEM/DSM heating system coefficients – for internal use only
tCEFFd	N	Y	Y	N	N	SBEM/DSM cooling system coefficients – for internal use only
SBEMHVACSystems	Y	N	N	N	N	SBEM HVAC system database – for internal use only
SBEMLightingTypes	Y	N	N	N	N	SBEM lighting system database – for internal use only
EnergyPlusVersions	Y	Y	Y	N	N	EnergyPlus versions list – for internal use only
SBEMVersions	Y	Y	Y	N	N	SBEM versions list – for internal use only
SBEMHeatSourceFuelType	N	Y	Y	N	N	SBEM data
SBEMRecirculation	N	Y	Y	N	N	SBEM data
SBEMHeatRecovery	N	Y	Y	N	N	SBEM data
SBEMHeatRecoverySystemTypes	N	Y	Y	N	N	SBEM data
SBEMDHWEfficiencies	N	Y	Y	N	N	SBEM data
SBEMDHWFuels	N	Y	Y	N	N	SBEM data
SBEMConstructionLibrary	N	Y	Y	N	N	SBEM data
SBEMConstructionInference	N	Y	Y	N	N	SBEM data
SBEMGlazingInference	N	Y	Y	N	N	SBEM data
SBEMGlazingLibrary	N	Y	Y	N	N	SBEM data
SBEMFrameLibrary	N	Y	Y	N	N	SBEM data
SBEMFrameInference	N	Y	Y	N	N	SBEM data
SBEMRecommendations	N	Y	Y	N	N	SBEM data
NewCFDAnalysis	Y	Y	Y	Y	N	New CFD analysis dialog
IEEGrausdia	Y	Y	Y	Y	N	Not used
OPTCalcOptions	Y	Y	Y	Y	N	Optimisation, Parametric analysis and UA/SA calculation options dialog
CFDBoundaryConditions	Y	Y	Y	Y	N	CFD boundary conditions dialog
SubBuildings	Y	N	N	N	N	Not used
Transport	Y	N	N	N	N	Not used
FormatTest	Y	N	N	N	N	Not used
RMCalcOptions	Y	Y	Y	Y	N	Daylighting calculation options dialog
HeatingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
CoolingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
FanComponent	Y	N	N	Y	Y	Detailed HVAC dialog
PumpComponent	Y	N	N	Y	Y	Detailed HVAC dialog
Boiler	Y	N	N	Y	Y	Detailed HVAC dialog
ChillerEIR	Y	N	N	Y	Y	Detailed HVAC dialog
DirectAirADU	Y	N	N	Y	Y	Detailed HVAC dialog
SeriesPIUADU	Y	N	N	Y	Y	Detailed HVAC dialog
ParallelPIUADU	Y	N	N	Y	Y	Detailed HVAC dialog
FourPipeInductionADU	Y	N	N	Y	Y	Detailed HVAC dialog
SingleDuctCAVReheatADU	Y	N	N	Y	Y	Detailed HVAC dialog

SingleDuctVAVReheatADU	Y	N	N	Y	Y	Detailed HVAC dialog
SteamHumidifier	Y	N	N	Y	Y	Detailed HVAC dialog
FanCoilUnit	Y	N	N	Y	Y	Detailed HVAC dialog
SingleDuctVAVReheatVSFanADU	Y	N	N	Y	Y	Detailed HVAC dialog
SingleDuctVAVNoReheatADU	Y	N	N	Y	Y	Detailed HVAC dialog
DualDuctCAVADU	Y	N	N	Y	Y	Detailed HVAC dialog
DualDuctVAVADU	Y	N	N	Y	Y	Detailed HVAC dialog
HVACZone	Y	N	N	Y	Y	Detailed HVAC dialog
HVACZoneGroup	Y	N	N	Y	Y	Detailed HVAC dialog
PlantLoop	Y	N	N	Y	Y	Detailed HVAC dialog
ChilledCeiling	Y	N	N	Y	Y	Detailed HVAC dialog
HeatedFloor	Y	N	N	Y	Y	Detailed HVAC dialog
CoolingTower	Y	N	N	Y	Y	Detailed HVAC dialog
AirLoop	Y	N	N	Y	Y	Detailed HVAC dialog
AirHandlingUnit	Y	N	N	Y	Y	Detailed HVAC dialog
Plenum	Y	N	N	Y	Y	Detailed HVAC dialog
CooledBeam	Y	N	N	Y	Y	Detailed HVAC dialog
SetpointManager	Y	N	N	Y	Y	Detailed HVAC dialog
CompactCurves	Y	N	N	N	N	Detailed HVAC dialog
CompactChillers	Y	N	N	N	N	Not used
CompactBoilers	Y	N	N	N	N	Not used
FrameConstruction	Y	N	N	N	N	
DXCoolingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
DXHeatingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
UnitaryHeatPump	Y	N	N	Y	Y	Detailed HVAC dialog
PackagedTerminalHeatPump	Y	N	N	Y	Y	Detailed HVAC dialog
DaylightDisplayOptions	Y	N	Y	Y	N	Daylight display options
WaterConvactor	Y	N	N	Y	Y	Detailed HVAC dialog
ElectricConvactor	Y	N	N	Y	Y	Detailed HVAC dialog
WaterRadiator	Y	N	N	Y	Y	Detailed HVAC dialog
ElectricRadiator	Y	N	N	Y	Y	Detailed HVAC dialog
PackagedTerminalAirConditioner	Y	N	N	Y	Y	Detailed HVAC dialog
UnitaryHeatCool	Y	N	N	Y	Y	Detailed HVAC dialog
WaterOutlet	Y	N	N	Y	Y	Detailed HVAC dialog
WaterOutletGroup	Y	N	N	Y	Y	Detailed HVAC dialog
WaterHeater	Y	N	N	Y	Y	Detailed HVAC dialog
RT2012DisplayOptions	Y	N	Y	N	N	RT2012 data
RT2012Versions	Y	Y	Y	N	N	RT2012 data
CFDComfortCalcOptions	Y	Y	Y	Y	N	CFD comfort calculation options dialog
PeoplePayback	Y	Y	Y	N	N	People payback calculator
CurrencyTemplates	Y	Y	Y	N	N	
RT2012Emetteurs	Y	N	Y	N	N	RT2012 data
RT2012VentMeca	Y	N	Y	N	N	RT2012 data
RT2012PuitsClimatique	Y	N	Y	N	N	RT2012 data
RT2012GenerateurCombustion	Y	N	Y	N	N	RT2012 data
LoadHVACTemplate	Y	Y	Y	N	N	Load Detailed HVAC template wizard
EvaporativeCooler	Y	N	N	Y	Y	Detailed HVAC
DetailedHVACTemplate	Y	N	Y	N	N	Detailed HVAC template data
SolarCollector	Y	N	N	Y	Y	
SolarCollectorTemplate	Y	N	N	N	N	
RT2012GenReseau	Y	N	Y	N	N	RT2012 data
OptimisationAnalysis	Y	N	Y	Y	N	Optimisation Analysis data
RT2012BallonsECS	Y	N	Y	N	N	RT2012 data
FangerComfort	Y	Y	Y	N	N	Fanger comfort dialog
OptimisationObjectives	Y	N	Y	N	N	Optimisation objective data
OptimisationConstraints	Y	N	Y	N	N	Optimisation constraint data
OptimisationVariables	Y	N	Y	N	N	Optimisation/parametric analysis and UA/SA variable data

Variables	N	Y	Y	N	N	Optimisation/parametric analysis and UA/SA variable options list
RT2012ThermodynElecNonRev	Y	N	Y	N	N	RT2012 data
RT2012SourceAmount	Y	N	Y	N	N	RT2012 data
HeatPumpHeating	Y	N	N	Y	Y	Detailed HVAC dialog
HeatPumpCooling	Y	N	N	Y	Y	Detailed HVAC dialog
GroundHeatExchanger	Y	N	N	Y	Y	Detailed HVAC dialog
PerformanceSimple	Y	N	N	N	N	Simple PV panel data
PerformanceOneDiode	Y	N	N	N	N	One diode PV panel data
Tariffs	Y	N	N	N	N	Tariff analysis data
Inverters	Y	N	N	N	N	Inverter data
Distributions	Y	N	N	N	N	Electrical distribution data
Storage	Y	N	N	N	N	Electrical storage (battery) data
RT2012ThermodynElecAut	Y	N	Y	N	N	RT2012 data
ComponentCost	Y	N	N	N	N	
LifeCycleCost	Y	N	N	N	N	Life cycle cost data
WindTurbine	Y	N	N	N	N	Wind turbine data
GHETemplate	Y	N	N	N	N	Detailed HVAC dialog
ZoneExhaustFan	Y	N	N	Y	Y	Detailed HVAC dialog
HPHeatingCoeffsParas	Y	N	N	N	N	Detailed HVAC dialog
HPCoolingCoeffsParas	Y	N	N	N	N	Detailed HVAC dialog
CasaClimaOptions	Y	Y	Y	Y	N	Not used
RT2012ThermodynGazNonRev	Y	N	Y	N	N	RT2012 data
CoolingSystems	Y	N	N	N	N	Cooling design Cooling systems data
GeometryConventionTemplates	Y	N	Y	N	N	Geometry conventions data
EMSPProgramManager	Y	N	N	Y	N	
EMSPPrograms	Y	N	N	N	N	
FMUPPrograms	Y	N	N	N	N	Not used
EMSInputData	Y	N	N	N	N	
SBEMSolarCollectors	Y	N	Y	N	N	SBEM solar collector data
VRFAirConditioner	Y	N	N	Y	Y	Detailed HVAC dialog
VRFTerminalUnit	Y	N	N	Y	Y	Detailed HVAC dialog
VRFDXCoolingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
VRFDXHeatingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
VRFAirConditionerTemplates	Y	N	N	N	N	Detailed HVAC dialog
VRFTUTemplates	Y	N	N	N	N	Detailed HVAC dialog
UnitaryWaterToAirHeatPump	Y	N	N	Y	Y	Detailed HVAC dialog
ZoneWaterToAirHeatPump	Y	N	N	Y	Y	Detailed HVAC dialog
WaterToAirHPDXCoolingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
WaterToAirHPDXHeatingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
MoistureMaterials	Y	N	N	N	N	Moisture materials database
FuelEmissionFactors	Y	N	N	N	N	
FluidToFluidHeatExchanger	Y	N	N	Y	Y	Detailed HVAC dialog
DistrictCooling	Y	N	N	Y	Y	Detailed HVAC dialog
DistrictHeating	Y	N	N	Y	Y	Detailed HVAC dialog
FluidCooler	Y	N	N	Y	Y	Detailed HVAC dialog
WaterHeaterHeatPump	Y	N	N	Y	Y	Detailed HVAC dialog
AirToWaterHeatPumpCoil	Y	N	N	Y	Y	Detailed HVAC dialog
GroundDomain	Y	N	N	N	N	Ground domain data
SummaryOutputs	Y	N	N	N	N	Parametric outputs list
FilterSelection	Y	Y	Y	N	N	
MergeModelData	Y	Y	Y	Y	N	
BaselineBuilding	Y	Y	Y	N	N	ASHRAE 90.1 baseline building wizard
ASHRAE901LPD	N	N	N	N	N	ASHRAE 90.1 data
ElectrochromicSensor	Y	N	N	N	N	Electrochromic sensor data
OrientationShadingCalcs	Y	Y	N	N	N	
CFDMeshCreation	Y	Y	Y	Y	N	CFD+ create mesh dialog
OutdoorAirVAVADU	Y	N	N	Y	Y	Detailed HVAC dialog

CostData	Y	Y	Y	Y	N	
DataPlotFieldIndexes	N	N	N	N	N	
ASHRAE901FloorDef	Y	N	N	N	N	ASHRAE 90.1 data
DIN41082	Y	Y	Y	Y	N	DIN4108.2 dialog
LEPOSTVariables	N	N	N	N	N	LEED MEPC report data
RT2012ThermodynGazDS	Y	N	Y	N	N	RT2012 data
RT2012VRVGeneration	Y	N	Y	N	N	RT2012 data
RT2012BoucleEau	Y	N	Y	N	N	RT2012 data
RT2012ThermodynGazRev	Y	N	Y	N	N	RT2012 data
RT2012ThermodynElecDS	Y	N	Y	N	N	RT2012 data
RT2012Supervision	Y	Y	Y	Y	N	RT2012 data
RadiantSurface	Y	N	N	Y	Y	Detailed HVAC dialog
LEPOSTAirSideVs	N	N	N	N	N	LEED MEPC data
ChilledWaterStorage	Y	N	N	Y	Y	Detailed HVAC dialog
IceThermalStorage	Y	N	N	Y	Y	Detailed HVAC dialog
CTThermalPerformance	Y	N	N	N	N	Detailed HVAC dialog
HVACGenerator	Y	N	N	Y	Y	Detailed HVAC dialog
OptimisationOutputs	Y	N	Y	N	N	Parametric outputs list
S63AltMeasures	Y	N	N	N	N	SBEM data
RT2012Ascenseur	Y	N	Y	N	N	RT2012 data
RT2012ListeParkings	N	N	Y	N	N	RT2012 data
RT2012Parking	Y	N	Y	N	N	RT2012 data
RT2012ListeAscenseurs	N	N	Y	N	N	RT2012 data
SBEMShowers	Y	N	N	N	N	SBEM data
PaneGroups	Y	N	N	N	N	Pane group database
InsertMaterialLayer	Y	N	N	N	N	Insert material dialog
AdvancedCFDErrorList	N	Y	Y	N	N	CFD+ error list
FoundationKivaInputs	Y	N	N	N	N	Kiva foundations database
FoundationKivaSettings	Y	N	N	N	N	Kiva settings database
ProjectDetails	Y	Y	Y	Y	N	SBEM/DSM project details dialog
EditThemes	Y	Y	Y	N	N	Themes data
HTRadiantHeating	Y	N	N	Y	Y	Detailed HVAC dialog
A901BuildingAreaTypes	Y	N	N	N	N	ASHRAE 90.1 data
UnitarySystem	Y	N	N	Y	Y	Detailed HVAC dialog
GenericCoolingCoil	Y	N	N	Y	Y	Detailed HVAC dialog
DefunctPanels	N	Y	Y	N	N	List of Panels no longer supported in IGDB

Columns

Editable

The “Editable” column indicates whether the table defines data and formats for a dialog. Tables with a “Y” in the Editable column are editable which means that as well as the data itself, an additional table is stored that describes the layout of the corresponding dialog.

The tables pointed to by records in TableOfTables are:

1. The table whose records provide the data itself. This table is based on the corresponding .dat file and can be accessed using the name indicated in the “Name” column of TableOfTables. This table is always present.
2. The corresponding formats table that defines the layout of the dialog. This table can be accessed by adding the text “Formats”. For example to access the formats table of the Constructions table the table name is “ConstructionsFormats”. This table is only present for editable tables.

This column corresponds to the “IsEditable” field in TableOfTables.

Application

The “Application” column indicates whether the table defines data associated the overall DesignBuilder application as opposed to the currently loaded model, or to the library data if no model is loaded. Tables with a “Y” in the Application column are associated with the application and those with an “N” are associated with the model, or with the library if no model is loaded.

This column corresponds to the “IsApplication” field in TableOfTables.

Template

The “Template” column indicates whether the table is for a template or component.

- a) Tables with a “Y” in the Template column are template tables and so when no model is loaded are accessed from the DB.Api.ApplicationTemplates class.
- b) Tables with a “N” in the Template column are component tables and so when no model is loaded are accessed from the DB.Api.ApplicationComponents class.

This column corresponds to the “IsTemplate” field in TableOfTables.

Model Data

The “Model Data” column indicates whether data associated with the table is accessed from model attributes or from a table accessed from model site level (when a model is loaded).

- a) Tables with a “Y” in the “Model Data” column access their data from model attributes when a model is loaded or from a table when no model is loaded. In this case, when a model is loaded the attribute names used to store the data are defined by the “Name” field in the fmt file and the level in the model where the data is loaded from is defined by the LowestDecompositionLevel field in the Formats.fmt model data layout table.
- b) Tables with a “N” in the “Model Data” column access their data from a table regardless of whether a model is loaded or not.

This column corresponds to the “LoadFromSketch” field in TableOfTables.

HVAC

The “HVAC” column indicates whether data associated with the table is associated with a Detailed HVAC component or not. “Y” indicates that it is and “N” indicates that it is not.

Note that HVAC tables only contain dialog layout data. The HVAC data itself is stored within the Detailed HVAC data structures.

This column corresponds to the “IsHVAC” field in TableOfTables.

Appendix 2- DesignBuilder Dialogs and Database Tables

DesignBuilder uses database tables to hold component and template data and also to define the layout of dialogs. This Appendix should ideally be read with reference to some example .dat and .fmt files open in a text editor.

DesignBuilder Data (.dat) file

e.g. Constructions.dat

Data (.dat) files contain the database records supplied with the installation of DesignBuilder. The user’s own custom records are appended to these and stored as part of their model (or as part of the library for library components and templates added on the opening screen). As with the format file, this is a text file with two lines of header information. The header is followed by a single line for each record in the database table. Each line (record) in the table is broken up into a number of fields using the # symbol as a delimiter.

Header

The first header line has one field for each of the “Categories” that subsequent records fall into. The content of each of these fields is the Id of the category found in the file Localised_Categories_1.txt.

The second line of the header is the list of Field names used by each record in the table.

Note: The first 3 fields in [editable](#) database tables must be the same as shown below.

Data

The remaining lines in the format file specify values for the records in the Database. Each line contains the data for one record.

Common fields to all database tables are:

Id – a unique Id for the record – DesignBuilder Database records have Ids in range 1-9999, User records have Ids in range 10000 – 99999.

Name – Unique user-defined name for the record.

CategoryId – The index into the list of categories defined on the first line of the .dat file.

The remaining fields are different for each database depending on the requirements of the corresponding component or template.

Table of Tables

The “Table of Tables” table is a special case of a data file. It is loaded from TableOfTables.dat and can be accessed through the API using code like:

```
Table = ApplicationTemplates.GetTable(“TableOfTables”)
```

It provides DesignBuilder with detailed information on each data table/dialog in the database.

You can find a copy of TableOfTables.dat in the DesignBuilder\Data folder which can be accessed from within DesignBuilder using the “File > Folders > Library data” folder menu command.

Note that [Appendix 1](#) includes a summary of the list of tables included in TableOfTables.

This table uses the standard format for DesignBuilder database tables with fields as follows:

Id - A unique Id for the table.

Name – The name of the table which is the same as the stem of the filename of the .dat & .fmt files.

CategoryId – Always set to 1 - only 1 category for this table i.e. All.

ParentNodekey – this is a unique three-letter code used to represent the table. E.g. ^^Con for Constructions.

CategoriesTableName -

Title – The Integer index into in LocalisedText_1.txt defines title of the table (use title case text, e.g. “Constructions”).

IconName – Stem of the filename of an Icon (bmp) file.

IsTemplate – Set to “True” if database table is a template object and “False” if a component.

IsApplicationData – Set to “False” if the data is stored in the dsb file, “True” if it is to be accessed at the application level or if data is .

UseGroupToolbox – Not used.

AllowNameCopy –

NameFieldName – The name of the “Name” field in the Database Table. Set to “Name” for all tables except LocationTemplates

PropertyMode – Set “True” if database values appear in the left-hand panel on the DesignBuilder main window, “False” if database values are edited in a pop-up dialog.

LoadFromSketch – Set “True” when data is to be read from model data attributes, “False” if the data is to be stored in a table (i.e. .dat file)

IsWizard – Is the dialog to be displayed in wizard form – normally “False”

AllowDontShowFormNextTime – When “True”, a checkbox is displayed at the bottom of the dialog to indicate that the dialog should not be displayed next time in a similar situation.

AlwaysAllowEdit – Set to “True” if the data can always be edited (e.g. program options dialog) or “False” if restrictions can apply (e.g. constructions dialog where the data may be library data and not editable).

IsDownloadable – A mechanism exists to allow a single .dat file to be shared by multiple users over the internet. This was implemented for the Messageboard (“Comments” table). Normally set to “False”.

IsEditable – “True” when a dialog exists (fmt file defined) or “False” for database tables without a dialog.

Width – The default width of the dialog used to edit the database values in twips.

InfoWidth – The default width of the pane at the RHS when “LearningMode” is on.

Height – Default height of dialog in twips.

InfoMinimised – “True” if the info panel on the right of the dialog is minimized by default

ShowInNavigators – “True” if the data in the table be shown in navigators.

IsListTable – “True” if the data a list of sub-items to be edited using a grid control (normally “False”)

FilterByRegion – When “True” the lists are filtered based on the region using the Region settings on the Program options dialog.

SingularName – What is the component called in its singular – integer identifier points to text in LocalisedText_1.txt.

NumberCurrentRegionData – Set to 0.

RemoveEmptyCategoryNodesWhenNotShowOtherRegionsData – Normally “False”, but set to “True” if empty category nodes in the navigator treeview should be removed. This will only be done when not showing all regions data as set on Program options dialog, International tab.

ShowInSBEMMode – “True” if the table should be displayed in navigator lists when the software is running in SBEM mode.

IsHVAC – “True” if the table defines the dialog layout for a Detailed HVAC component.

DesignBuilder Table Format Files

Dialogs in DesignBuilder are mostly data-driven, meaning that the layout of the items on the dialog is defined using a “format file”. These format files are loaded into DesignBuilder tables at the time the software is built and at run time the software accesses the data from the tables and not from the format files.

Format files use the .fmt file extension, e.g. “Constructions.fmt”.

Most plugins do not need to access format files as they will include their own dialogs, however in certain cases it may be useful to understand how dialogs are created and in very unusual circumstances changes can be made by for example changing max/min or default values.

Third-party developers do not have direct access to the source .fmt files, but they can be accessed using the API as described above in [Appendix 1](#) and, if necessary, regenerated using code like:

```
Table.SaveAsTextFile FMTFileName
```

The format file defines the layout of the items on the dialog. It is a text file with two lines of header information followed by a single line for each item displayed on the dialog. These items include database fields and items to control the layout of the dialog. Each line in the fmt file is broken up into a number of fields using the # symbol as a delimiter.

The first header line has one field for each of the tabs to be displayed on the dialog. The content of each of these fields is the Id of the title of the tab found in LocalisedText_1.txt.

The second line of the header is the list of identifiers for the fields in the remaining lines of the file. This second line should be the same for all formats.

The remaining lines in the format file specify the various controls shown on the dialog.

Each line has the following fields:

Name – A Name for the control on the dialog, if the control is used to edit a DesignBuilder Database field then the name should match the name of the corresponding field in the .dat file.

Caption – The integer index of the string used for the caption of the control. The index points into LocalisedText_n.txt which contains the actual string. Strings for the various supported languages come from these files depending on the language selected on the program options dialog:

- LocalisedText_1.txt – English
- LocalisedText_2.txt - Italian
- LocalisedText_3.txt - Spanish
- LocalisedText_4.txt - French

Default – The default value for newly created data. May be left blank. For Header data this value determines whether the header is open or closed by default. 1 means closed, 0 means open.

Indent – The level of indentation. If a control is part of a group of controls that only displayed when a checkbox is checked or a header is expanded then their indentation should be set to one greater than the “parent” checkbox or header item.

OptionGroup – used for linking radio buttons (ItemType = GreyOptionBox) to define the other radio button controls that should be cleared when this one is selected.

ItemType – An integer defining the type of control:

ItemType	Comments
0 – Header	Grey header can be opened or closed – no corresponding field required in .dat file for Header item types.
1 – GreyCheckBox	Standard check box – when it is checked, child items are shown.
2 – GreyOptionBox	Radio button – see also info under OptionGroup above.
3 – Browse	In this case the table from which the data is to be selected should be entered in ListType.
4 – Combo	List of sequential integers defined in ListType
5 – Button	Not commonly used
7 – Text or Numeric edit	Numeric data requires format, max and min values. If the data is non-numeric make sure to set the min and max values to 0.
8 – Colour	A colour can be selected and displayed
9 – Label	Non-editable data

10 – TextCombo	Requires the list index in LocalisedLists_1.txt to defined in subsequent ListType field
11 – Object	e.g. special control such as graph, picture etc
12 – Slider	Slider controls need to have a (unique within the table) ObjectIndex integer defined as well.
13 – Profile Edit	In this case make sure to define a unique ObjectIndex integer.
14 – Option Box	not used.
15 – Check Box	not used.
16 – File Browse	Browse for a file.
17 – Path Browse.	Browse for a path.
18 – Spring	Not used
19 – Multi-Select Browse	Allows multiple items to be selected from a database table.
20 – Multi-line edit	Multi-line editor for recording notes etc
21 – Date	Enter a date
22 – Grid	A grid control
23 – Encrypted Filename Browse	Only for textures
22 – Grid	A grid control
30 – File Browse	Full Path
31 – File Browse (EPW)	Browse for a weather file
77 – Edit (Numeric or Autosize)	Like 7, 77 supports numeric input but also allows 'Autosize' or other text input

Labels – The labels to be used for Slider or ProfileEdit controls. In this case the data in ListType (next field) is the minor increment for the slider or profile edit control.

ListType – Has a different meaning depending on ItemType:

ItemType	Meaning of data in ListType
3 (Browse)	Table name from which the item can be
4 (Combo)	3 integers separated by commas defines the start number, followed by the last number followed by the step. e.g. 1,10,1 generates a list of 10 integers from 1 to 10.
10 (TextCombo)	Selected Integer Id of a list of values from Localised_Lists

Enabled – Whether the control is enabled or not. Normally set to “True”

OrderIt – Not used

Lowest Decomposition Level – Used when the [LoadFromSketch](#) TableOfTable value is set to “True”. This is the decomposition level at which the data is loaded/stored from the model data.

- 0=site
- 1=building
- 2=block
- 3=zone
- 4=surface
- 5=opening

Format – A string specifying the number of decimal places should be displayed (e.g. 0.000) indicates 3 decimal places.

Locked – Set to “True” if user is not allowed to edit the data.

Min - Sets minimum permitted value for numeric controls. Set to 0 for non-numeric data.

Max – Sets maximum permitted value for numeric controls. Set to 0 for non-numeric data.

ObjectIndex – The index of the slider or ProfileEdit control.

ValueLabel – Used for ProfileEdit controls.

TabIndex – Sets which of the dialog tabs the control is to be displayed on. The first tab on the dialog has index 1.

Id – Used internally by DB, set to 0.

Units – For numeric data, this is Id of the Units for the value from DesignBuilder “Units” table (Units.dat). Leave it blank for non-numeric data, or 99 for numeric but with no units (factors etc).

ShowNCM – Indicates whether the data is shown when DB is running in SBEM mode. 1 for yes, 0 for no.

ShowEnergyPlus – Indicates whether the data is shown when DB is running in EnergyPlus mode. 1 for yes, 0 for no.

ShowKLIMA – Indicates whether the data is shown when DB is running in Klima mode. 1 for yes, 0 for no.

ShowPT – Indicates whether the data is shown when in PT IEE mode. 1 for yes, 0 for no (Not used).

ShowES - Indicates whether the data is shown when DB is running in Spanish mode (Not Used)

ShowDE - Indicates whether the data is shown when DB is running in German mode (Not Used)

ShowFR – Indicates whether the data is shown when using the RT2012 Analysis type.

ShowDBSim – Indicates whether the data is shown when running in DBSim mode (Not Used)

ShowL5 - Indicates whether the data is shown when running in DSM mode.

The first few lines and fields of Constructions.fmt file are shown below to illustrate the layout of fmt files. Note the use of “#” characters as a field delimiter.

Line 1 is list of tabs to be displayed on the dialog. Each number is an index into the LocalisedText_1.txt file

Line 2 is the list of fields, the same for every fmt file

Lines 3 onwards are the format records with each line defining a control on the dialog

1	#4601	#7292	#4602	#4603	#9202	#9257	#10116	#11496	#	#Enabled	#OrderIt	#LowestDecompositionLevel	#Format	#Locked
2	#Name	#Caption	#Default	#Indent	#OptionGroup	#ItemType	#Labels	#ListType						
3	#General	#2566	#	#0	#0	#0	#	#		#True	#0	#0	#	#False
4	#Name	#2567	#New construction	#1	#0	#7	#	#		#True	#10	#0	#	#False
5	#Source	#2570	#	#1	#0	#7	#	#		#True	#-5000	#0	#	#False
6	#CategoryId	#2568	#1	#1	#0	#3	#	#ConstructionCategories		#True	#11	#0	#	#False
7	#RegionId	#2569	#315	#1	#0	#3	#	#Regions		#True	#316	#0	#	#False
8	#Colour	#3422	#12632256	#1	#0	#8	#	#Colour		#True	#50	#0	#	#False
9	#DefinitionMethodHead	#14861	#	#0	#0	#0	#	#		#True	#0	#0	#	#False
10	#DefinitionMethod	#5660	#1	#1	#0	#10	#	#346		#True	#0	#0	#	#False
11	#CFactorHead	#13089	#	#0	#0	#0	#	#		#True	#-10000	#0	#	#False
12	#CFactor	#13087	#0.99	#1	#0	#7	#	#		#True	#10	#0	#0.000	#False
13	#FFactorHead	#13090	#	#0	#0	#0	#	#		#True	#-10000	#0	#	#True
14	#FFactor	#13088	#1.264	#1	#0	#7	#	#		#True	#10	#0	#0.000	#False
15	#LibraryInferenceHead	#5669	#	#0	#0	#0	#	#		#True	#-10000	#0	#	#True
16	#LibraryCategory	#5663	#0	#1	#0	#10	#	#		#True	#0	#0	#	#False
17	#LibraryConstructionName	#5664	#0	#1	#0	#10	#	#		#True	#0	#0	#	#False
18	#InferenceSector	#5665	#0	#1	#0	#10	#	#		#True	#0	#0	#	#False
19	#InferenceRegs	#5666	#0	#1	#0	#10	#	#		#True	#0	#0	#	#False
20	#InferenceGD	#5667	#0	#1	#0	#10	#	#		#True	#0	#0	#	#False
21	#TypeHead	#5517	#	#0	#0	#0	#	#		#True	#0	#0	#	#False
22	#SBEMType	#5516	#1-Exterior	#1	#0	#10	#	#333		#True	#115	#0	#	#False
23	#SettingsHeader	#11219	#1	#0	#0	#0	#	#		#True	#0	#0	#	#False
24	#SimulationAlgorithmOverride	#11220	#1-Default	#1	#0	#10	#	#1019		#True	#0	#0	#	#False
25	#InvolvesMetalCladding	#5480	#0	#1	#0	#1	#	#		#True	#0	#0	#	#False
26	#L5GroundFloorUValueCorrected	#6426	#0	#1	#0	#1	#	#		#True	#0	#0	#	#False
27	#LayersHead	#2571	#	#0	#0	#0	#	#		#True	#0	#0	#	#False
28	#NumberLayers	#2572	#1	#1	#0	#4	#	#1,10,1		#True	#115	#0	#0	#False
29	#Layer 1	#2575	#	#1	#0	#0	#	#		#True	#60	#0	#	#False
30	#Mat 1	#2576	#1	#2	#0	#3	#	#Materials		#True	#3345	#0	#	#False
31	#Thick 1	#2577	#0.2	#2	#0	#7	#	#		#True	#117	#0	#0.0000	#False
32	#Bridged 1	#2578	#0	#2	#0	#1	#	#		#True	#115	#0	#	#False
33	#BMat 1	#2576	#1	#3	#0	#3	#	#Materials		#True	#1	#0	#	#False
34	#PercBridge 1	#2579	#0	#3	#0	#7	#	#		#True	#117	#0	#	#False
35	#PVPanelHeader	#16602	#0	#2	#0	#0	#	#		#True	#115	#0	#	#False

DesignBuilder Edit Screen Format file

The model data Edit Format data has 2 purposes:

1. To define the attributes used in DesignBuilder models
2. To define the layout of the UI items on the model data tabs, similar to pop-up dialogs.

The model data format file is loaded into DesignBuilder at the time the software is built and can be accessed using the EditFormats table from the ApplicationTemplates object (see code snippet below).

The source format file is called "Formats.fmt".

Most plugins do not need to access the model data format file as they will add their own dialogs, however in certain circumstances changes can be made. For example, it may be useful to change max/min or default values or even add new attributes and UI items. Also, it can be useful to obtain a full list of the model data attributes.

Tip: A sample plugin called **DBUIAdditionPluginExample** is provided in the [GitHub repository](#). It illustrates how to add new custom attributes and corresponding UI items.

Third-party developers do not have direct access to the source Formats.fmt file, but the data can be accessed using the API as described above in [Appendix 1](#) and, if necessary, regenerated using code as follows:

```
FormatsTable = ApiEnvironment.ApplicationTemplates.GetTable("EditFormats")
FormatsTable.SaveAsTextFile "Formats.fmt"
```

Formats.fmt file is a text file with two lines of header information followed by a single line for each item displayed on the dialog. These items include database fields and items to control the layout of the dialog. Each line is broken up into a number of fields using the # symbol as a delimiter.

The layout of Formats.fmt is similar to the other table format files described in the previous [DesignBuilder Table Format Files](#) section so to avoid repetition only the exceptions are described here. Where no description is provided please refer to the previous section.

Name

Caption

IsData – Defines whether this item includes data (an attribute) or not. 1 if it is data, or 0 if it is not. For example header controls do not hold data and so have a value of 0.

Default

Indent

OptionGroup

ItemType

Labels

ListType

Enabled

DataType – Not used

LowestDecompositionLevel – the level down to which the attribute data is inherited.

Format

Min

Max**Locked**

SiteTab – The index of the tab on which this control is displayed when at site level. A value of -1 means that the data is not displayed at site level.

The tabs are listed in Tabs.fmt. You can find a copy of the this file in the DesignBuilder\Data folder which can be accessed from within DesignBuilder using the “File > Folders > Library data” folder menu command.

BuildingTab – The index of the tab on which this control is displayed when at building level. A value of -1 means that the data is not displayed at building level.

PBTab – The index of the tab on which this control is displayed when at block level. A value of -1 means that the data is not displayed at block level.

ZoneTab – The index of the tab on which this control is displayed when at zone level. A value of -1 means that the data is not displayed at zone level.

SurfaceTab – The index of the tab on which this control is displayed when at surface level. A value of -1 means that the data is not displayed at surface level.

OpeningTab – The index of the tab on which this control is displayed when at opening level. A value of -1 means that the data is not displayed at opening level.

ObjectIndex**ValueLabel**

Visible – True if this control is to be displayed, False if not.

ParametricList – for ProfileEdit controls this is the list of parametric options.

UniqueCaption – caption to be used for parametric simulation results.

IconName – name of the icon to be included in the control.

ChangeUpdatesRenderedView – 1 if changes to this attribute should result in the model being updated through a call to the UpdateTileAttributes API method, otherwise 0. UpdateTileAttributes regenerates default opening arrangements and rebuilds the OpenGL display using the latest attribute settings.

ChangeUpdatesShading – 1 if changes to this attribute should result in the shading results view being reset, otherwise 0. Not Used.

HideWhenMerged - 1 if this attribute should be hidden for child merged zones, otherwise 0.

ChangeUpdatesHG – 1 if changes to this attribute should result in the Cooling design results being deleted, otherwise 0. Not Used.

ChangeUpdatesSS – 1 if changes to this attribute should result in the Simulation results being deleted, otherwise 0. Not Used.

ChangeUpdatesGeometry – 1 if changes to this attribute should result in the API UpdateGeometry method being called, otherwise 0. UpdateGeometry regenerates zone inner volumes and the OpenGL display.

MyTab – The index of the tab on which this control is displayed. -1 if there is no UI item associated with this control.

Id – For internal use.

Dirty – For internal use.

TemplateIdName – The name of the Template attribute Id which is used to load data for this attribute. For example, if the attribute is loaded from Constructions Templates then this should have the value ConstructionTemplateId. Leave it blank if the data is not loaded from a template.

Prefix – Not used.

Units

ShowNCM

ShowEnergyPlus

ShowKLIMA

ShowPT

ShowES

ShowDE

ShowFR

ShowDBSim

ShowL5

The first few lines and fields of Formats.fmt file are shown below to illustrate the layout. Note the use of “#” characters as a field delimiter.

Line 1 is not used

Line 2 is the list of format fields

Lines 3 onwards are the format records with each line defining a control on the Edit screen

#	Name	#Caption	#IsData	#Default	#Indent	#OptionGroup	#ItemType	#Labels
2	#Name	#Caption	#IsData	#Default	#Indent	#OptionGroup	#ItemType	#Labels
3	#TariffMainHeader	#12016	#0	#	#0	#0	#0	#
4	#TariffIncluded	#12448	#1	#0	#1	#0	#1	#
5	#TariffCount	#12017	#1	#2	#2	#0	#4	#
6	#TariffId1	#12018	#1	#1	#2	#0	#3	#
7	#TariffId2	#12139	#1	#2	#2	#0	#3	#
8	#TariffId3	#12140	#1	#3	#2	#0	#3	#
9	#TariffId4	#12141	#1	#4	#2	#0	#3	#
10	#TariffId5	#12142	#1	#5	#2	#0	#3	#
11	#ComponentCostMainHeader	#11360	#0	#	#0	#0	#0	#
12	#ComponentCostIncluded	#12449	#1	#0	#1	#0	#1	#
13	#ComponentCostSettings	#12179	#1	#1	#2	#0	#3	#
14	#LifeCycleCostMainHeader	#12180	#0	#	#0	#0	#0	#
15	#LifeCycleCostIncluded	#12450	#1	#0	#1	#0	#1	#
16	#LifeCycleCostSettings	#12180	#1	#1	#2	#0	#3	#
17	#ConstructionTemplateHeader	#4094	#0	#	#0	#0	#0	#
18	#ConstructionTemplateId	#4095	#1	#1	#1	#0	#3	#
19	#Insulation	#4096	#1	#3	#1	#0	#12	#267
20	#ThermalMass	#4097	#1	#1	#1	#0	#12	#268
21	#ConstructionHead	#4100	#0	#	#0	#0	#0	#
22	#SelectiveZoneMergingData	#7261	#1	#	#1	#0	#7	#
23	#Title	#4101	#1	#Title	#1	#0	#7	#
24	#RTWallAreaNotAvailableForWindows	#16324	#1	#0	#1	#0	#1	#
25	#WallConstr	#4102	#1	#0	#1	#0	#3	#
26	#WallBelowGradeConstr	#12451	#1	#0	#1	#0	#3	#
27	#CombinedFlatRoofConstr	#5064	#1	#1755	#1	#0	#3	#
28	#PitchedRoofConstr	#4106	#1	#0	#1	#0	#3	#
29	#UnoccupiedPitchedRoofConstr	#4744	#1	#1205	#1	#0	#3	#
30	#InternalWallConstr	#4103	#1	#0	#1	#0	#3	#
31	#SemiExposedHead	#5078	#0	#	#1	#0	#0	#
32	#SemiExposedWallConstr	#4104	#1	#1007	#2	#0	#3	#
33	#CombinedSemiExposedRoofConstr	#5071	#1	#1731	#2	#0	#3	#
34	#CombinedSemiExposedFloorConstr	#5072	#1	#1719	#2	#0	#3	#
35	#Floors	#4113	#0	#	#1	#0	#0	#
36	#CombinedGroundFloorConstr	#5066	#1	#1743	#2	#0	#3	#
37	#BasementGroundFloorConstr	#13092	#1	#1743	#2	#0	#3	#
38	#CombinedExternalFloorConstr	#5068	#1	#1707	#2	#0	#3	#
39	#CombinedInternalFloorConstr	#5065	#1	#84	#2	#0	#3	#

Model Attributes

The available model attributes are listed below. These can be useful when writing EMS, CS-Script and Python scripts or plugins.

Tip: It is usually more convenient to access the names of the attributes that appear in the user interface by switching on the [Show attribute names in tooltips](#) Program option.

Attribute List

TariffIncluded	SBEMPVSIInclination	SSTemperatureDistributionOP
TariffCount	SBEMPVSOvershadeType	SSSaveSurfaceOP
TariffId1	SBEMPVSVentStrategy	SSSaveOpeningOP
TariffId2	RTVegetationSurMasque	SSAirflowInOP
TariffId3	RTUsageVegetation	SSAirflowOutOP
TariffId4	RTDebutVegetation	SSSolarTransOP
TariffId5	RTDebutVegetationJour	SSIntSurfaceTempOP
ComponentCostIncluded	RTDebutVegetationMois	SSExtSurfaceTempOP
ComponentCostSettings	RTFinVegetation	SSIntConvectionCoeffOP
LifeCycleCostIncluded	RTFinVegetationJour	SSExtConvectionCoeffOP
LifeCycleCostSettings	RTFinVegetationMois	SSSurfaceGainOP
ConstructionTemplateId	CFDNonOrthoVentBoundaryElementWidth	SSSolarIncidentOP
Insulation	CFDNonOrthoVentBoundaryElementHeight	SSExtSunlitFractionOP
ThermalMass	CFDNonOrthoVentBoundaryFaceOffset	SSComfortHead
SelectiveZoneMergingData	CFDNonOrthoVentBoundaryXSpacing	SSFangerComfortOP
Title	CFDNonOrthoVentBoundaryYSpacing	SSPierceComfortOP
RTWallAreaNotAvailableForWindows	CFDNonOrthoVentBoundaryXEdgeOffset	SSKSUComfortOP
WallConstr	CFDNonOrthoVentBoundaryYEdgeOffset	SSAdaptiveASH55ComfortOP
WallBelowGradeConstr	CFDInternalSurfaceTemperature	SSAdaptiveCEN15251ComfortOP
CombinedFlatRoofConstr	CFDExternalSurfaceTemperature	SSCIBSETM52ComfortOP
PitchedRoofConstr	CFDExternalWindowTemperature	SSCIBSETM52SuggestedAcceptableRange
UnoccupiedPitchedRoofConstr	CFDBoundaryMoistureFlux	SSCIBSETM59ComfortOP
InternalWallConstr	CFDBoundaryMoistureFluxPerUnitArea	SSCIBSETM59MechVentOP
SemiExposedWallConstr	CFDOpeningFlowIn	SSCIBSETM59VulnerableOccOP
CombinedSemiExposedRoofConstr	CFDOpeningFlowOut	SSCIBSETM59CorridorOP
CombinedSemiExposedFloorConstr	CFDOpeningAdjacentTemperature	SSCIBSETM59Vent
CombinedGroundFloorConstr	CFDBoundaryContaminantMassFraction	SSSimpleASH55ComfortOP
BasementGroundFloorConstr	CFDBoundaryContaminantFluxPerUnitArea	OutputSummaryHeadSS
CombinedExternalFloorConstr	CFDBoundaryRelativeHumidity	SSSummaryOutputUnits
CombinedInternalFloorConstr	CFDComponentBlockThermalBoundary	AllSummary
RTPontThermIntermediaire	CFDComponentBlockTemperature	LEEDSummary
RTTIntRefendTrav	CFDComponentBlockFlux	AnnualBuildingUtilityPerformanceSummary
RTPTIntPsiRefendTrav	CFDComponentContaminantFlux	DemandEndUseComponentsSummary
RTPTIntLongRefendTrav	CFDComponentMoistureFlux	SensibleHeatGainSummary
RTTIntRefendBas	CFDComponentBlockMass	InputVerificationandResultsSummary
RTPTIntPsiRefendBas	CFDSolidCellOverlapTolerance	SourceEnergyEndUseComponentsSummary
RTPTIntLongRefendBas	CFDVentElementDimensionOrthoSurfaces	AdaptiveComfortSummary
RTTIntRefendHaut	CFDVentElementDimensionNonOrthoSurfaces	ZoneComponentLoadSummary
RTPTIntPsiRefendHaut	CFDComponentBlockFlowRate	Standard621Summary
RTPTIntLongRefendHaut	CFDComponentBlockFlowDirection	EnergyMeters
RTTIntPoutre	CFDOpeningAperturePosition	ClimaticDataSummary
RTPTIntPsiPoutre	CFDOpeningPercAperture	EquipmentSummary
RTPTIntLongPoutre	CFDBoundaryType	EnvelopeSummary
SubSurfaceConstr	CFDBoundaryPressure	SurfaceShadowingSummary
InternalSubSurfaceConstr	CFDBoundaryPressureType	ShadingSummary
RoofSubSurfaceConstr	CFDBoundaryTemperature	LightingSummary
ExternalDoorConstr	CFDBoundaryFlowRate	HVACSizingSummary
InternalDoorConstr	CFDBoundaryXDirectionDischargeAngle	SystemSummary
HasSolarCollector	CFDBoundaryYDirectionDischargeAngle	ComponentSizingSummary
SBEMSolarCollector	CFDBoundaryMultiWayDiffuserDischargeAngle	OutdoorAirSummary
SBEMSolarCollectorArea	CFDBoundaryTwoWayDiffuserDischargeDirection	ObjectCountSummary
SBEMSolarCollectorPerc	CFDBoundaryMinimumDischargeVelocity	ComponentCostEconomicsSummary
ThermalMassConstr	CFDBoundaryActualVelocity	OutputMonthlyHeadSS
ThermalMassArea	CFDBoundaryFlux	AIMMonthly
ZoneCapacitanceMultiplier	CFDBoundaryH2OMassFraction	ZoneHeatingSummaryMonthly
RTTypeInertie	CFDBoundaryCO2MassFraction	ZoneCoolingSummaryMonthly
RTTypeInertieSeq	CFDSurfaceRefinementLevel	ZoneElectricSummaryMonthly
RTCmsSurf	CFDAssemblyBoxRefinementRegionExtent	SpaceGainsMonthly
RTAmsSurf	CFDRefinementRegionLevel4	PeakSpaceGainsMonthly
RTTypeInertieAnnuel	CFDRefinementRegionLevel4Distance	SpaceGainComponentsAtCoolingPeakMonthly
RTCmqSurf	CFDRefinementRegionLevel3	EnergyConsumptionElectricityNaturalGasMonthly
RTAmqSurf	CFDRefinementRegionLevel3Distance	EnergyConsumptionElectricityGeneratedPropaneMonthly
ComponentBlockType	CFDRefinementRegionLevel2	EnergyConsumptionDieselFuelOilMonthly
SolarCollectorType	CFDRefinementRegionLevel2Distance	EnergyConsumptionDistrictHeatingCoolingMonthly
SolarCollectorDepth	CFDRefinementRegionLevel1	EnergyConsumptionCoalGasolineMonthly
SolarCollectorCost	CFDRefinementRegionLevel1Distance	EndUseEnergyConsumptionElectricityMonthly
ComponentBlockShades	SSEPObjectNameInOP	EndUseEnergyConsumptionNaturalGasMonthly
ComponentBlockEPLLevel	IDFName	EndUseEnergyConsumptionDieselMonthly
ComponentMaterial	ObjectTag	EndUseEnergyConsumptionFuelOilMonthly
ComponentShadingSurface	HLIncludeUnoccupiedZones	EndUseEnergyConsumptionCoalMonthly
ComponentMaxTransmittance	HLSaveSurfaceOP	EndUseEnergyConsumptionPropaneMonthly
ComponentTransmittanceSchedule	HLSaveOpeningOP	EndUseEnergyConsumptionGasolineMonthly
PVPerformanceType	HGIncludeUnoccupiedZones	EndUseEnergyConsumptionOtherFuelsMonthly
PVPerformanceModelSimple	HGSaveSurfaceOP	PeakEnergyEndUseElectricityPart1Monthly
PVPerformanceModelOneDiode	HGSaveOpeningOP	PeakEnergyEndUseElectricityPart2Monthly
PVHeatTeansferIntegrationMode	SSEPObjectAreaInOP	ElectricComponentsOfPeakDemandMonthly
PVModulesInSeries	SSEPObjectDCInOP	PeakEnergyEndUseNaturalGasMonthly
PVStringsInParallel	SSBuildingStoreyPerimeterBlockOP	PeakEnergyEndUseDieselMonthly
SBEMPVSON	SSIncludeUnoccupiedZones	PeakEnergyEndUseFuelOilMonthly
SBEMPVSMultiplier	SSAllowCustomOutputs	PeakEnergyEndUseCoalMonthly
SBEMPVSUseAreaType	SSSurfaceOP	PeakEnergyEndUsePropaneMonthly
SBEMPVSArea	SSTemperatureOP	PeakEnergyEndUseGasolineMonthly
SBEMPVSWattPeakType	SSGainsOP	PeakEnergyEndUseOtherFuelsMonthly
SBEMPVSUseKWPeak	SSEnergyOP	SetpointsNotMetWithTemperaturesMonthly
SBEMPVSKWPeak	SSLatentOP	ComfortReportSimple55Monthly
SBEMPVSOrientation	SSFreshAirOP	UnglazedTranspiredSolarCollectorSummaryMonthly

OccupantComfortDataSummaryMonthly	JunctWallFir	RTModeGestionMiSaison
ChillerReportMonthly	JunctLintel	RTModeGestionEte
TowerReportMonthly	JunctSill	RTModeGestionEteClimatise
BoilerReportMonthly	JunctJamb	RTModeGestionCalculTic
DXReportMonthly	RTIsDetailCalcul	RTValeurDeclareeDefault
WindowReportMonthly	RTIfloorWallOrientation	RTDeltaThetaOp1Aut
WindowEnergyReportMonthly	RTRoofWallOrientation	RTSaisonChauaffeDeltaThetaOp1
WindowZoneSummaryMonthly	RTWallWallOrientation	RTMiSaisonDeltaThetaOp1
WindowEnergyZoneSummaryMonthly	RTOriginePsiLiaison	RTSaisonRefrroidDeltaThetaOp1
AverageOutdoorConditionsMonthly	RTJustifArt19b	RTCalculTicDeltaThetaOp1
OutdoorConditionsMaximumDryBulbMonthly	L5PsiAcc	RTDeltaThetaOp2Aut
OutdoorConditionsMinimumDryBulbMonthly	L5SpecifyPsi	RTSaisonChauaffeDeltaThetaOp2
OutdoorConditionsMaximumWetBulbMonthly	L5JunctRoofWallIMC	RTMiSaisonDeltaThetaOp2
OutdoorConditionsMaximumDewPointMonthly	L5JunctWallGrndMC	RTSaisonRefrroidDeltaThetaOp2
OutdoorGroundConditionsMonthly	L5JunctWallWallIMC	RTCalculTicDeltaThetaOp2
WindowACReportMonthly	L5JunctWallFirMC	RTDeltaThetaAut
WaterHeaterReportMonthly	L5JunctWallExtFirMC	RTSaisonChauaffeDif
GeneratorReportMonthly	L5JunctLintelMC	RTMiSaisonDif
DaylightingReportMonthly	L5JunctSillMC	RTSaisonRefrroidDif
CoilReportMonthly	L5JunctJambMC	RTCalculTicDif
PlantLoopDemandReportMonthly	L5JunctRoofWall	RTThetaEiSeuilAut
FanReportMonthly	L5JunctWallGrnd	RTSaisonChauaffeMin
PumpReportMonthly	L5JunctWallWall	RTMiSaisonMin
CondLoopDemandReportMonthly	L5JunctWallFir	RTSaisonRefrroidMin
ZoneTemperatureOscillationReportMonthly	L5JunctWallExtFir	RTCalculTicMin
AirLoopSystemEnergyAndWaterUseMonthly	L5JunctLintel	RTThetaEiSeuilHautAut
AirLoopSystemComponentLoadsMonthly	L5JunctSill	RTSaisonChauaffeMax
AirLoopSystemComponentEnergyUseMonthly	L5JunctJamb	RTMiSaisonMax
MechanicalVentilationLoadsMonthly	ThermalBridgeAngularThreshold	RTSaisonRefrroidMax
OutputDaylightHeadSS	RTPTSFilCRoofWall	RTCalculTicMax
SSDaylightingMapOP	RTPTSFilERoofWall	RTTThetaOpBaseAut
SSDaylightingMapOPGridPoints	RTPTSFilCWaIlGrnd	RTSaisonChauaffeOp
SSDaylightReferencePointsOP	RTPTSFilEWaIlGrnd	RTMiSaisonOp
SSSurfaceNodeTemperatures	RTPTSFilCWaIlWall	RTSaisonRefrroidOp
SSSystemTemperatures	RTPTSFilEWaIlWall	RTCalculTicOp
SSSystemMassFlowRates	RTPTSFilCWaIlFir	RTSsurventilation
SSSystemVolumeFlowRateStandard	RTPTSFilEWaIlFir	RTDeltaTravSurv
SSSystemVolumeFlowRateCurrent	RTPTSFilCWaIlExtFir	RTVentNatET
SSSystemHumidityRatios	RTPTSFilEWaIlExtFir	RTHTftf
SSSQLiteOP	RTPTSFilCLintel	RTDefQvETbase
SSDXFOP	RTPTSFilELintel	RTQvETbase
SSConstructionsOP	RTPTSFilCSill	GlazingType
SSRDDOP	RTPTSFilESill	FacadeTemplated
RMWorkingPlaneHeight	RTPTSFilCJamb	RTExpBR
RTMydatecCoche	RTPTSFilEJamb	Layout
RTMydatecCepCh	RTJunctMurDebord	PercGlazing
RTMydatecCepFr	RTPTSFilCMurDebord	PreferredFrameWidth
RTMydatecCepVentil	RTPTSFilEMurDebord	PreferredFrameHeight
RTMydatecAepenr	RTPTLongueurMurDebord	WindowSpacing
RTTwidoCoche	RTPTSoustraireDebordPlanchInt	SillHeight
RTTwidoCepECS	RTJunctWallPartition	RevealOutsideProjection
AdjacentCondition	RTPTLongueurWallPartition	SurfaceAreaRatio
SBEMIsStronglyVentilated	RTPTSFilCWaIlPartition	HasFrame
SBEMSameSpace	RTPTSFilEWaIlPartition	WindowFrameType
RolPartiallyConditioned	RTJunctFloorPartition	RevealInsideProjection
S63LoftSpaceAccessible	RTPTLongueurFloorPartition	InsideSillDepth
IsExcludedFromZoneFloorArea	RTPTSFilCFloorPartition	FrameOuterWidth
GeometryConventionTemplateId	RTPTSFilEFloorPartition	FrameInsideProjection
ZoneSurfaceAreaCalculationMethod	RTJunctRoofPartition	FrameOutsideProjection
ZoneVolumeCalculationMethod	RTPTLongueurRoofPartition	FrameConductanceRatio
ZoneFloorAreaCalculationMethod	RTPTSFilCRoofPartition	DividerType
WindowToWallRatioSurface	RTPTSFilERoofPartition	FrameInnerWidth
FixedExternalWallThickness	InfiltrationOn	HorizontalDividers
ExternalWallThickness	InfiltrationValue	VerticalDividers
FixedExternalWallBelowGradeThickness	UKNCMQ50INF	DividerOutsideProjection
ExternalWallBelowGradeThickness	RTStatutQ4PaSurf	DividerInsideProjection
FixedPartitionThickness	InfiltrationValueI4	DividerConductanceRatio
PartitionThickness	InfiltrationValuen50	SBEMSHADINGSYSTEM
FixedGroundFloorThickness	InfiltrationOverrideEI	SBEMSHADINGPOSITION
GroundFloorThickness	InfiltrationEI	SBEMSHADINGCOLOUR
FixedBasementGroundFloorThickness	InfiltrationOverrideEpsilon	SBEMSHADINGTX
BasementGroundFloorThickness	InfiltrationEpsilon	ShadingCoeff
FixedExternalFloorThickness	InfiltrationSchedule	SBEMBRISESOLEIL
ExternalFloorThickness	InfiltrationConstantTermCoef	InternalBlinds
FixedInternalFloorThickness	InfiltrationTempTermCoef	InternalBlind
InternalFloorThickness	InfiltrationVelocityTermCoef	InternalBlindPosition
FixedSemiExposedWallThickness	InfiltrationVelocitySquareTermCoef	InternalBlindControl
SemiExposedWallThickness	SealingQuality	InternalBlindControlMultipleStateElectrochromic
FixedSemiExposedFloorThickness	CrackTemplate	InternalBlindOverrideGlare
SemiExposedFloorThickness	NaturalVentedCavityOn	InternalBlindGlareOverrideIncidentSetpoint
FixedCeilingThickness	NaturalVentedCavityFractionOfOpenings	InternalBlindGlareOverrideLowAltitudeSetpoint
CeilingThickness	NaturalVentedCavityThermalEmissivityOfBaffle	InternalBlindGlareOverrideHighAltitudeSetpoint
FixedSemiExposedRoofThickness	NaturalVentedCavitySolarAbsorptivityOfBaffle	InternalBlindControlPriorityMode
SemiExposedRoofThickness	NaturalVentedCavityHeightScaleForBuoyancyVentilation	InternalBlindDaylightControlOverride
FixedFlatRoofThickness	NaturalVentedCavityEffectiveThicknessOfCavityBehindBaffle	InternalBlindControlSolar
FlatRoofThickness	NaturalVentedCavityRatioOfActualToProjectedSurfaceArea	InternalBlindControlOutsideAirTemp
FixedPitchedRoofThickness	NaturalVentedCavityRoughnessOfExteriorSurface	InternalBlindControlInsideAirTemp
PitchedRoofThickness	NaturalVentedCavityEffectivenessForPerforations	IsElectrochromicSensor
CeilingVoidDepth	NaturalVentedCavityDischargeCoefficientForOpenings	ElectrochromicSensorObject
FloorVoidDepth	UseGroundDomainFloor	InternalBlindSlatAngleControl
HLEPInsideConvectionAlgo	UseGroundDomainWall	InternalBlindSlatAngleSchedule
HLEPOutsideConvectionAlgo	SelectGroundDomain	InternalBlindControlScheduleDefinition
HGEPInsideConvectionAlgo	UseKivaSurface	InternalBlindControlSchedule
HGEPOutsideConvectionAlgo	KivaSurfaceInputs	InternalBlindControlEP
SSEPInsideConvectionAlgo	SubStructureCostPerFloorArea	InternalBlindControlControl
SSEPOutsideConvectionAlgo	FrameType	LouvresOn
SBEMSpecifyPsi	CostInternalWallSurfaceFinish	LouvreType
JunctRoofWallIMC	CostFloorFinish	AirflowControlWindowsOn
JunctWallGrndMC	CostCeilingFinish	AirflowControlWindowsSource
JunctWallWallIMC	RTExigenceMoyConstrEntete	AirflowControlWindowsDestination
JunctWallFirMC	RTArt18	AirflowControlWindowsMaxFlowRate
JunctLintelMC	GlazingTemplated	AirflowControlWindowsScheduleDefinition
JunctSillMC	RTOuvertureBaies	AirflowControlWindowsSchedule
JunctJambMC	RTGestionOuvertureFenetre	RTValeurSaisieDefaultRouvmax
JunctRoofWall	RTFenetreNonOuvrable	RTTypeOuvrant
JunctWallGrnd	RTGestOuvertureFenetre	ExtWinCOMISScheduleDefinition
JunctWallWall	RTModeGestionHiver	ExtWinCOMISSchedule

ExtWinNaturalVentilationEP	VentHeightAboveFloor	RTT5LAHERoofEnteteConstr
ExtWinNaturalVentilationControl	RTDebitVentNat	RTT5LAHERoofTypeComble
ExtWinFreeAperture	RTQvBase	RTOccupationPassagere
ExtWinOpeningPosition	RTQvPointe	RTEstLocalSommeil
ExtWinNaturalVentilationPercOpeningValue	RTQvInocc	RTNBlogement
ExtWinOpeningCd	RTQvOcc	RTTransfertAir
InternalGlazingType	RTVentNatDepuisET	RTIsHail
InternalFacadeTemplateId	RTIdEtAeration	RTIsTraversant
InternalLayout	ExtVentCOMISScheduleDefinition	RTPEBNEnteteActive
InternalPercGlazing	ExtVentCOMISSchedule	RTPEBNAvecAscenseurs
InternalPreferredFrameWidth	ExtVentNaturalVentilationEP	RTPEBNAvecParkings
InternalPreferredFrameHeight	ExtVentNaturalVentilationControl	RTPEBNModeSaisieParkings
InternalWindowSpacing	InternalVentType	RTPEBNnbPlacesParkings
InternalSillHeight	InternalVentsOn	RTPEBNModeSaisieSurfParkings
InternalHasFrame	InternalVentArea	RTPEBNSurfParkings
InternalWindowFrameType	InternalVentSpacing	RTPEBNTypeEclParkings
InternalHorizontalDividers	InternalVentHeightAboveFloor	RTPEBNTypeVentilParkings
InternalVerticalDividers	NVIntVentControlOption	RTSurfacesRegl
InternalFrameOuterWidth	IntVentCOMISScheduleDefinition	RTSHONRT
InternalFrameInnerWidth	IntVentCOMISSchedule	RTSHONRTDelta
NVIntWinControlOption	IntVentNaturalVentilationEP	RTSURT
IntWinCOMISScheduleDefinition	IntVentNaturalVentilationControl	RTSURTDelta
IntWinCOMISSchedule	RoofVentType	RTSHAB
IntWinNaturalVentilationEP	RoofVentsOn	RTSHABDelta
IntWinNaturalVentilationControl	RoofVentArea	RTIsRatl
IntWinFreeAperture	RoofVentSpacing	RTRatl
IntWinOpeningPosition	RoofVentHeightAboveFloor	RTCategorieEntete
IntWinNaturalVentilationPercOpeningValue	RTEntreAir	RTCategorieCe1Ce2
IntWinOpeningCd	RTPresenceEntreAir	ASHRAE901Type
RoofGlazingType	RTTypeEntreeAir	BuildingConstructionStage
RoofFacadeTemplateId	RTModule	BuildingNature
SBEMRoofLightCover	RTModule1	IncludeExistingBuildingInTheModel
RoofLayout	RTModule2	PrimaryBuildingConditionCategory
RoofPreferredFrameWidth	RTDeltaP1	ASHRAE901HeatingSource
RoofPreferredFrameHeight	RTDeltaP2	ASHRAE901LightingCategory2007
RoofWindowSpacing	RTAttARegul	ASHRAE901LightingCategory2010
RoofSillHeight	RTExigencesHabitation	ASHRAE901FloorDefinition
RoofRevealOutsideProjection	RTArt20Entete	BuildingOrientation
RoofSurfaceAreaRatio	RTArt20Calcule	ConditionedFloorArea
RoofHasFrame	RTArt20Manuel	ConditionedVolume
RoofWindowFrameType	RTArt20	UnconditionedFloorArea
RoofHorizontalDividers	AllGainsUnits	UnconditionedVolume
RoofVerticalDividers	AllGainsValue	OccupiedFloorArea
RoofFrameOuterWidth	AllGainsAbsoluteValue	OccupiedVolume
RoofFrameInnerWidth	AllGainsSchedule	UnoccupiedFloorArea
RoofSBEMSHADINGSYSTEM	AllGainsEP	UnoccupiedVolume
RoofSBEMSHADINGPOSITION	EquipmentList	FloorArea
RoofSBEMSHADINGCOLOUR	IEEZoneType	ZoneVolume
RoofSBEMSHADINGTX	ActivityTemplateId	ZoneVolumeEdit
RoofShadingCoeff	DetailedOccupantId	InnerSurfaceMode
RoofInternalBlinds	SectorId	OccupancyMethod
RoofInternalBlind	SubBuildingId	OccupancyFloorAreaPerPersonIsOccupied
RoofInternalBlindPosition	ZoneType	OccupancyFloorAreaPerPerson
RoofInternalBlindControl	ZoneUnconditionedVentilated	OccupancyValuesOccupied
RoofInternalBlindControlMultipleStateElectrochromic	ASHRAE901SpaceConditionCategory	OccupancyValue
RoofInternalBlindOverrideIfGlare	IEEUnoccupiedZoneType	OccupancyNumberOfPeopleIsOccupied
RoofInternalBlindGlareOverrideIncidentSetpoint	ZoneMultiplier	OccupancyNumberOfPeople
RoofInternalBlindGlareOverrideLowAltitudeSetpoint	EquipmentUnits	OccupancySchedule
RoofInternalBlindGlareOverrideHighAltitudeSetpoint	FavouredOrientation	OccupancyEP
RoofInternalBlindControlPriorityMode	IncludeZone	OccupancyDaysWeek
RoofInternalBlindDaylightControlOverride	IncludeZoneDaylighting	OccupancyLatentGainMethod
RoofInternalBlindControlSolar	RTInclureZoneRT	MetabolicRateld
RoofInternalBlindControlOutsideAirTemp	SetZoneHeight	MetabolicFactor
RoofInternalBlindControlInsideAirTemp	ZoneHeightIs	CarbonDioxideGenerationRate
RoofIsElectrochromicSensor	FitOutArea	OccupancyLatentFraction
RoofElectrochromicSensorObject	NullZone	CloScheduleDef
RoofInternalBlindSlatAngleControl	RTTestGroupe	WinterClo
RoofInternalBlindSlatAngleSchedule	RTZoneRTduGroupe	SummerClo
RoofInternalBlindControlScheduleDefinition	RTLocalAvecEclairageSpecifique	CloSchedule
RoofInternalBlindControlSchedule	RTUsageEntete	MRTCalcType
RoofInternalBlindControlEP	RTUsage	AirVelocityComfortCalculationSched
RoofInternalBlindControlControl	RTLieuxResidentielCollectif	IncludeContaminant
RTValeurSaisieDefaultRouvmaxRoof	RTLieuxBureau	ContaminantSimulationMethod
RTTypeOuvrantRoof	RTLieuxCommerceMagasinZI	ConstantContaminantMaxGenerationRate
ExtRoofCOMISScheduleDefinition	RTLieuxCrecheGarderiePouponieres	CutoffContaminantDesignGenerationRate
ExtRoofCOMISSchedule	RTLieuxEnseignementPrimaire	DecaySourceContaminantInitialGenerationRate
ExtRoofNaturalVentilationEP	RTLieuxEtabSanitaireAvecHeberg	ContaminantGenerationSchedule
ExtRoofNaturalVentilationControl	RTLieuxEtabSportifMunicipalPrive	CutoffContaminantCutoffConcentration
ExtRoofFreeAperture	RTLieuxEtablissementSportifScolaire	DecaySourceContaminantDecayTimeConstant
ExtRoofOpeningPosition	RTLieuxHebergementOccContinue	ConstantContaminantDesignRemovalCoefficient
ExtRoofNaturalVentilationPercOpeningValue	RTLieuxHebergementResidenceEtudiante	DepositionRateSinkContaminantDepositionRate
ExtRoofOpeningCd	RTLieuxHopitalpartiejour	ContaminantRemovalSchedule
SBEMDoorType	RTLieuxHopitalpartienuit	UseHolidays
DoorsOn	RTLieuxHotel1ou2etoilesJour	DaysHolidays
DoorPreferredWidth	RTLieuxHotel1etoileNuit	HolidaysId
DoorPreferredHeight	RTLieuxHotel2etoileNuit	DHWVValue
ExtDoorOpenPerc	RTLieuxHotel3ou4etoilesJour	RTECNbMaisonGrE
ExtDoorPercTimeOpen	RTLieuxHotel3etoileNuit	RTECNuGrEmeLit
ExtDoorOpeningPosition	RTLieuxHotel4etoileNuit	RTECNuGrEmeSurface
ExtDoorCOMISScheduleDefinition	RTLieuxIndustrie38	RTECNuGrEmeChambre
ExtDoorCOMISSchedule	RTLieuxIndustrie8h18	RTECNuGrEmeRepas
ExtDoorNaturalVentilationEP	RTLieuxRestaurant1Repas57	RTECNuGrEmeDouches
ExtDoorNaturalVentilationControl	RTLieuxRestaurant2Repasjour67	HeatingDesignSetPointTemperature
InternalDoorsOn	RTLieuxRestaurant2Repasjour77	HeatingSetBackSetPointTemperature
InternalDoorPreferredWidth	RTLieuxRestaurant3Repasjour57	ActiveCoolingDesignSetPointTemperature
InternalDoorPreferredHeight	RTLieuxRestaurationContinue	CoolingSetBackSetPointTemperature
IntDoorOpenPerc	RTLieuxSecondaireJour	HeatingPMVSetPoint
IntDoorPercTimeOpen	RTLieuxSecondaireNuit	HeatingPMVSetBack
IntDoorOpeningPosition	RTLieuxTransportAerogare	CoolingPMVSetPoint
IntDoorCOMISScheduleDefinition	RTLieuxTribunal	CoolingPMVSetBack
IntDoorCOMISSchedule	RTLieuxUniversite	HumidificationSP
IntDoorNaturalVentilationEP	RTUvueMI	DehumidificationSP
IntDoorNaturalVentilationControl	RTUvueLC	IncludeNatVentIndoorMinTemp
VentType	RTUvueTer	NatVentIndoorMinTempDefinition
VentsOn	RTQueSaisi	NatVentSetPointTemperature
VentArea	RTRESETenteteActive	NatVentIndoorMinTempSchedule
VentSpacing	RTNatureLNC	IncludeNatVentIndoorMaxTemp

NatVentIndoorMaxTempDefinition	LightingControlOn	EconomiserUpperEnthalpy
NatVentIndoorMaxTempValue	WorkingPlaneHeight	HeatRecoveryHead
NatVentIndoorMaxTempSchedule	LightingControlType	HeatRecoveryOn
MechVentSetPointTemperature	LightingControlSteps	HeatRecoveryType
MechVentTemperatureDifference	LightingControlContinuousMinOPFrac	SensibleHeatRecoveryEffectiveness
MinFreshAir	LightingControlContinuousMinIPPowerFrac	LatentHeatRecoveryEffectiveness
MinFreshAirPerArea	LightingControlContinuousOffMinOPFrac	HeatRecoveryHeatingSP
CO2SetPoint	LightingControlContinuousOffMinIPPowerFrac	HeatRecoverySchedule
CO2MinConcentration	MaxGlareIndex	HasSupplyPlenumZone
CO2MaxConcentration	ViewAzAngle	SupplyPlenumLocation
ContaminantSetPoint	LightingZone1Perc	SupplyPlenumZone
ContaminantMinConcentration	LightingZone2On	HasReturnPlenumZone
RTMinFreshAirInnocc	TargetIlluminance2	ReturnPlenumLocation
RTMinFreshAirPerAreaInnocc	LightingZone2Perc	ReturnPlenumZone
TargetIlluminance1	SBEMPARASITICPOWER	VAVHead
LIGHTINGDISPLAY	SBEMLIGHTPARASPW	SupplyAirMinFlowFraction
ComputersOn	SBEMAUTOSUBDIVIDE	VAVType
ComputersUnits	SBEMMANUALSUBDIVIDE	ParallelFanOnFlowFraction
ComputersValue	SBEMMANUALSUBDIVIDEPERC	VAVFanHead
ComputersAbsoluteValue	SBEMLIGHTOCCSENS	VAVFanPressureRise
ComputersEP	SBEMLIGHTOCCSENS	VAVFanEff
ComputersSchedule	SBEMLIGHTOCCSENSParasiticPower	NCMLocalMechanicalSupply
ComputersRadiantFraction	TungstenLightingOn	NCMSpecificSupplyFanPowerUnknown
EquipmentOn	SBEMLIGHTTIMESWITCH	NCMDDefaultSpecificSupplyFanPower
OfficeEquipmentUnits	LightTimeSwitchSBEM4	NCMSpecificSupplyFanPowerKnown
EquipmentValue	SBEMLIGHTDISPHOFF	NCMSpecificSupplyFanPower
EquipmentAbsoluteValue	SBEMLIGHTDISPF	NCMDemandCtrlVentilation
EquipmentEP	SBEMDISPLAYEFF	NCMDemandCtrlVentFlowCtrl
EquipmentSchedule	SBEMLAMPBALLASTEFFDL	NCMZonalSupplyHeatRecoveryOn
EquipmentRadiantFraction	TungstenLightingUnits	NCMZonalSupplyHeatRecoveryType
MiscellaneousOn	TungstenLightingAbsoluteValue	NCMZonalHeatRecoverySEFFVariable
MiscellaneousUnits	TungstenLightingValue	NCMZonalHeatRecoveryUnknown
MiscellaneousValue	TungstenLightingEP	NCMZonalHeatRecoveryDefaultSEFF
MiscellaneousAbsoluteValue	TungstenLightingSchedule	NCMZonalHeatRecoveryKnown
MiscellaneousEP	TungstenLightingEndUseSubcategory	NCMZonalHeatRecoverySEFF
MiscellaneousSchedule	ExteriorLightingOn	NCMUnKnownSpecificFanPowerTU
MiscellaneousFuel	ExteriorLightingValue	NCMDDefaultSpecificFanPowerTU
MiscellaneousFractionLost	ExteriorLightingEP	NCMKnowSpecificFanPowerTU
MiscellaneousLatentFraction	ExteriorLightingSchedule	NCMSpecificFanPowerTU
MiscellaneousRadiantFraction	ExteriorLightingControlOption	NCMLocalMechanicalExhaust
MiscellaneousCO2GenRate	NCMLightingMandT	NCMLocalMechanicalExhaustScope
MiscellaneousEndUseSubcategory	NCMLightingMandTWithAlarmForOutOfRangeValues	NCMLocalMechanicalExhaustFlow
CateringOn	NCMLightingVoltageReduction	NCMUnKnownSpecificExtractFanPower
CateringUnits	LENICalc2	NCMDDefaultSpecificExtractFanPower
CateringValue	LightingCostPerFloorArea	NCMKnowSpecificExtractFanPower
CateringAbsoluteValue	RTExigenceMoyEclEnteteHab	NCMSpecificExtractFanPower
CateringEP	RTArt27	NCMUnKnownHEPA
CateringSchedule	RTArt28	NCMKnowHEPA
CateringFuel	RTExigenceMoyEclEnteteNonHab	HEPADEF
CateringFractionLost	RTArt37	NCMAirConInstalled
CateringLatentFraction	RTArt38	NCMAirConInspected
CateringRadiantFraction	RTArt39Entete	NCMAirConRatedOutputUnknown
CateringCO2GenRate	RTArt39a	NCMEstimatedtAirConRatedOutput
CateringEndUseSubcategory	RTArt39b	NCMAirConRatedOutputKnown
ProcessOn	RTArt40Entete	NCMAirConRatedOutput
ProcessUnits	RTArt40	NCMNightCooling
ProcessValue	RTArt41	NCMNightCoolingMaxFlow
ProcessAbsoluteValue	HVACTemplateId	NCMNightCoolingMaxHours
ProcessSchedule	SBEMZoneHasHVAC	NCMNightCoolingSFP
ProcessEP	SBEMUsesDefault	tAEnergy
ProcessFuel	SBEMDefaultHVAC	PumpEtcEnergy
ProcessFractionLost	SBEMDefaultHVACFuelType	PumpEtcScheduleDefinition
ProcessLatentFraction	SBEMKnownHVAC	PumpEtcSchedule
ProcessRadiantFraction	SBEMHVACSystemId	PumpEtcEP
ProcessCO2GenRate	NCMDestratificationFans	PumpEtcControl
ProcessEndUseSubcategory	EarlyHVACType	PumpEtcDaysWeek
LightingTemplateId	HVACAvailabilitySchedule	HeatingOn
LightingUnits	HVACAvailabilityEP	NCMHeatSource
ASHRAE901LightingSystemExistance	HVACAvailabilityControl	NCMECA
SpaceUseClassification	HVACAvailabilityDaysWeek	RTTypePgrmCh
FluorescentLightingOn	NightCycleControl	Boiler
SBEMLIGHTCASE	IsControlZone	BoilerFuel
FluorescentLightingUnits	IsMixedModeControlZone	BoilerCoP
FluorescentLightingValue	IsHumidificationControlZone	HeatingDistributionLoss
FluorescentLightingWattsPerM2Per100Lux	IsDehumidificationControlZone	SCoPFrac
FluorescentLightingAbsoluteWatts	ASHRAE901PriBaselineHVACSystemDesc	NCMBBoilerDate
RTPeclAux	ASHRAE901MajorProposedHVACSystemDesc	PreheatType
RTEffLampe	MechanicalVentilationOn	PreheatCoilSetpointTemp
RTRatLocalAuto	MechanicalVentilationRateType	CentralHeatingCoilType
RTRatLocal	MechanicalVentilationValue	HeatingCoilSetpointResetType
RTControleEcl	MechanicalVentilationDesignFlowRate	HeatingCoilSetpointTemp
RTGestEcl	MaxSupplyAirFlowRate	HeatingCoilMinOutdoorTemp
RTGradEcl	MinOutsideAirFlowRate	HeatingCoilSetpointTempAtMaxOutdoorTemp
RTRatioEclNat	RTQvPointeCVC	HeatingCoilMaxOutdoorTemp
RTFrGradEcl	RTQvBaseCVC	AHUHeatingCoilsSchedule
NCMTargetIlluminance	RTPiecesLogement	AHUHeatingCoilsEP
NCMDesignIlluminance	MechanicalVentilationSchedule	AHUHeatingCoilsPreheat
SBEMLAMPBALLASTEFF	MechanicalVentilationEP	AHUHeatingCoilsControl
SBEMLIGHTOUTPUTRATIO	MechanicalVentilationControl	AHUHeatingCoilsDaysWeek
SBEMLIGHTTYPE	MechanicalVentilationDaysWeek	ReheatType
SBEMDesignIlluminanceKnown	OutsideAirMixingHead	ZoneDamperHeatingAction
NCMDesignIlluminance2	FreshAirType	HeatingMaxLoad
SBEMAIREXTRACTINGLUMINAIRES	OutsideAirControlType	HeatingMaxLoadOP
FluorescentLightingSchedule	FanHead	HeatingMechVentSizingType
FluorescentLightingEP	FanType	HeatingMechVentSizingSystemSupplyTemperature
LuminaireType	FanMode	ExcludeHeatRecoveryHL
LightingReturnAirFraction	FanPressureRise	HeatingNatVentSizingType
LightingRadiantFraction	FanEff	HeatingType
LightingVisibleFraction	FanInAir	HeatingRF
LightingConvectiveFraction	FanPlacement	RadiantDistribution
SBEMCONSTANTILLUMINANCE	FanPartLoadPowerCoefficients	HeatingPASupplyAirTemperature
SBEMLIGHTCONTROLMANUAL	Economiser	HeatingPASupplyAirHumidityRatio
SBEMLIGHTCONTROLPHOTO	EconomiserType	IdealLoadsHeatingLimitType
L5OpposingWindows	EconomiserMaxACH	BaseboardOn
SBEMLIGHTPHBACK	EconomiserLockout	BaseboardType
SBEMLIGHTPHOTOTYPE	EconomiserUpperTemperature	BaseboardMode
SBEMPHOTOTYPE	EconomiserLowerTemperature	BaseboardMaxLoad

BaseboardRadiantFraction	NatVentMixedModeMinTemperature	OtherServicesCostPerFloorArea
BoilerSchedule	NatVentMixedModeMaxTemperature	GeneratorsYesNo
BoilerEP	NatVentMixedModeMinEnthalpy	DistributionCount
Preheat	NatVentMixedModeMaxEnthalpy	DistributionId1
BoilerControl	NatVentMixedModeMinDewPointTemperature	DistributionId2
BoilerDaysWeek	NatVentMixedModeMaxDewPointTemperature	DistributionId3
ActiveCoolingOn	NatVentMixedModeControlModeScheduleId	DistributionId4
CoolingSystemId	NatVentMixedModeMinOutdoorVentilationAirScheduleId	DistributionId5
CoolingMaxLoad	NatVentMixedModeAirflowNetworkControlTypeScheduleId	RTExigenceMoyEnteteHab
CoolingMaxLoadOP	NatVentMixedModeSimpleAirflowControlTypeScheduleId	RTArt23
HGCoolingMaxLoadMonth	RTAscenseursOn	RTArt24
RTTypePgrmFr	RTListeAscenseurs	RTArt25
Chiller	RTParkingsOn	RTArt26
ChilledWaterSetpointTemp	RTListeParkings	RTArt29
CondenserWaterSetpointTemp	EarthTubeYesNo	RTExigenceMoyEnteteNonHab
CoolingTowerType	EarthTubeRateType	RTArt31Entete
ChillerFuel	EarthTubeValue	RTArt31Chaud
SSEERFrac	EarthTubeDesignFlowRate	RTArt31Froid
CoolingPASupplyAirTemperature	EarthTubeSchedule	RTArt31ECS
CoolingPASupplyAirHumidityRatio	EarthTubeMinZoneTemp	RTArt31Eclairage
IdealLoadsCoolingLimitType	EarthTubeMaxZoneTemp	RTArt31Reseau
ChillerCoP	EarthTubeDeltaTemp	RTArt31Vent
CoolingType	EarthTubeType	RTArt3180Am
CoolingRF	EarthTubeFanPressureRise	RTArtVentEntete
NCMPackChillerType	EarthTubeFanTotalEfficiency	RTArt30v
NCMPackChillerkW	EarthTubePipeRadius	RTArt32
CondenserType	EarthTubePipeThickness	RTArt33
CoolingDistributionLoss	EarthTubePipeLength	RTArt42
CentralCoolingCoilType	EarthTubePipeThermalConduct	RTArtChauEntete
CoolingCoilSetpointResetType	EarthTubePipeDepthUnderGroundSurface	RTArt34
CoolingCoilSetpointTemp	EarthTubeSoilCondition	RTArt35
CoolingCoilMaxOutdoorTemp	EarthTubeAverageSoilSurfTemp	RTArtDistChaudFroidEntete
CoolingCoilSetpointTempAtMinOutdoorTemp	EarthTubeAmplitudeSoilSurfTemp	RTArt36
CoolingCoilMinOutdoorTemp	EarthTubePhaseConstantSoilSurfTemp	RTArtFroidEntete
AHUCoolingCoilsSchedule	EarthTubeConstantTermFlowCoef	RTArt43
AHUCoolingCoilsEP	EarthTubeTempTermFlowCoef	RTArt44
AHUCoolingCoilsPrecool	EarthTubeVelocityTermFlowCoef	RTArt45
AHUCoolingCoilsControl	EarthTubeVelocitySquareTermFlowCoef	TopologyType
AHUCoolingCoilsDaysWeek	SurfaceForm	DrawingObject
ChillerEP	WindAngle0	CuttingMode
ChillerSchedule	WindAngle45	IsGable
Precool	WindAngle90	IsOccupied
ChillerControl	WindAngle135	RoofOverlap
ChillerDaysWeek	WindAngle180	RoofSlope
HumidificationOn	WindAngle225	WallSlope
HumidificationAvailabilitySchedule	WindAngle270	Height
HumidificationElectricPower	WindAngle315	AutoHeightRipple
HumidificationCapacity	AirDistributionMode	lwwSelectionIndex
SimpleHVACHumidificationControlType	AirDistributionInterpolationMode	Length
DehumidificationOn	AirDistributionCouplingStrategy	LengthA
DehumidificationControlType	AirDistributionDesignTemperature	LengthB
SimpleHVACDehumidificationControlType	AirDistributionDesignHeatRate	Width
SimpleHVACCoolingSensibleHeatRatio	AirDistributionDesignTemperatureGradient	WidthA
DHWOn	AirDistributionReferenceTemperature	WidthB
DHWTemplatId	AirDistributionReferenceHeatRate	Depth
DHWDeadlegLength	AirDistributionReferenceTemperatureGradient	NumberStories
DHWType	AirDistributionSchedule	StoreyHeight
DHWCoP	AirDistributionEP	PercPartyWalls
DHWFuelType	AirDistributionControl	PitchedRoof
DHWIsPost98	AirDistributionThermometerHeight	LocationTemplatId
DHWStorageOn	AirDistributionReturnHeight	RTDepartement
DHWStorageVolume	AirDistributionRASGainDistributionSchedule	SBEMLocation
DHWStorageLoss	AirDistributionRASPlumesPerOccupant	TitleSite
DHWStorageLoss_kWhDay	AirDistributionRASNumberOfDiffusers	Latitude
DHWSecondaryCirculationOn	AirDistributionRASPowerPerPlume	Longitude
DHWSecondaryCirculationLoopLength	AirDistributionRASDesignAreaOfDiffuser	ClimaticRegion
DHWSecondaryCirculationLoss	AirDistributionRASDiffuserSlotAngle	Elevation
DHWSecondaryCirculationPumpPower	AirDistributionRASDiffuserType	IEESeaDistance
DHWOutputTemperature	AirDistributionRASCoefficientA	IEEConcelhos
DHWInputTemperature	AirDistributionRASCoefficientB	Exposure
DHWSchedule	AirDistributionRASCoefficientC	SiteOrientation
DHWEF	AirDistributionRASCoefficientD	UseSiteHeightVar
NaturalVentilationOn	AirDistributionRASCoefficientE	HeightVarWindSpeedProfileExponent
NaturalVentilationRateType	AirDistributionRASThermostatHeight	HeightVarWindSpeedProfileBoundaryLayerThickness
NaturalVentilationValue	AirDistributionRASComfortHeight	HeightVarTemperatureGradientCoefficient
NaturalVentilationDesignFlowRate	AirDistributionRASSetTransitionHeight	RTSiteProtege
NaturalVentilationSchedule	AirDistributionRASTransitionHeight	RTZoneEleIntLit
NaturalVentilationEP	AirDistributionRASReportingThreshold	RTConductiviteSol
NaturalVentilationDaysWeek	SBEMCHPOn	UseGroundConstr
IncludeNatVentOutdoorMinTemp	SBEMCHPHEATSEER	GroundConstr
NatVentOutdoorMinTempDefinition	SBEMCHPSHEAT	GroundTexture
NatVentOutdoorMinTempValue	SBEMCHPDHEAT	GroundReflectance
NatVentOutdoorMinTempSchedule	SBEMCHPHEATPOWERRATIO	GroundSnowReflectedSolarModifier
IncludeNatVentOutdoorMaxTemp	SBEMCHPPQAI	GroundSnowReflectedDaylightModifier
NatVentOutdoorMaxTempDefinition	SBEMCHPFUELTYPE	IncludeGroundDomains
NatVentOutdoorMaxTempValue	SBEMCHPTriGeneration	GroundModellingMechanism
NatVentOutdoorMaxTempSchedule	SBEMCHPSCOOOL	NumOfGroundDomains
IncludeNatVentDeltaTemp	SBEMCHPCOOOLSEER	GroundDomain1
NatVentDeltaTempDefinition	SBEMWINDGENERATOROn	GroundDomain2
NatVentTemperatureDifference	SBEMWINDGENERATORMultiplier	GroundDomain3
NatVentDeltaTempSchedule	SBEMWINDGENERATORArea	GroundDomain4
NatVentConstantTermCoef	SBEMWINDGENERATORHubHeight	GroundDomain5
NatVentTempTermCoef	SBEMWINDGENERATORPower	GroundDomain6
NatVentVelocityTermCoef	SBEMWINDGENERATORTerrainType	GroundDomain7
NatVentVelocitySquareTermCoef	L5AncillaryOn	GroundDomain8
COMISWindFactor	L5AncillaryPower	GroundDomain9
NatVentControlMode	RoIPrimaryEnergyExported	GroundDomain10
IntNatVentControlMode	RoIPrimaryEnergyUsed	KivaSettings
COMISModulationOn	EIFDistrictHeatingEfficiency	SimpleKivaSurfaceInput
COMISModulationMinDT	EIFDistrictCoolingCOP	JanGroundTemperature
COMISModulationMaxDT	EIFSteamConversionEfficiency	FebGroundTemperature
COMISModulationMaxDTFact	EIFTotalCarbonEquivalentEmissionFactorFromN2O	MarGroundTemperature
NatVentMixedModeOn	EIFTotalCarbonEquivalentEmissionFactorFromCH4	AprGroundTemperature
NatVentMixedModeOpeningFactorFunctionWindSpeedCurveId	EIFTotalCarbonEquivalentEmissionFactorFromCO2	MayGroundTemperature
NatVentMixedModeSwitchOffWhenRaining	HVACCostPerFloorArea	JunGroundTemperature
NatVentMixedModeMaxWindSpeed	HVACCostPerHeatingLoad	JulGroundTemperature
	HVACCostPerCoolingLoad	AugGroundTemperature

SepGroundTemperature
 OctGroundTemperature
 NovGroundTemperature
 DecGroundTemperature
 JanDeepGroundTemperature
 FebDeepGroundTemperature
 MarDeepGroundTemperature
 AprDeepGroundTemperature
 MayDeepGroundTemperature
 JunDeepGroundTemperature
 JulDeepGroundTemperature
 AugDeepGroundTemperature
 SepDeepGroundTemperature
 OctDeepGroundTemperature
 NovDeepGroundTemperature
 DecDeepGroundTemperature
 JanShallowGroundTemperature
 FebShallowGroundTemperature
 MarShallowGroundTemperature
 AprShallowGroundTemperature
 MayShallowGroundTemperature
 JunShallowGroundTemperature
 JulShallowGroundTemperature
 AugShallowGroundTemperature
 SepShallowGroundTemperature
 OctShallowGroundTemperature
 NovShallowGroundTemperature
 DecShallowGroundTemperature
 FCfactorMethodUseWeatherDataControl
 JanFCfactorMethodGroundTemperature
 FebFCfactorMethodGroundTemperature
 MarFCfactorMethodGroundTemperature
 AprFCfactorMethodGroundTemperature
 MayFCfactorMethodGroundTemperature
 JunFCfactorMethodGroundTemperature
 JulFCfactorMethodGroundTemperature
 AugFCfactorMethodGroundTemperature
 SepFCfactorMethodGroundTemperature
 OctFCfactorMethodGroundTemperature
 NovFCfactorMethodGroundTemperature
 DecFCfactorMethodGroundTemperature
 UseSkyTexture
 SkyTexture
 IncludeHorizon
 HorizonComponent
 SiteWaterMainsCalculationMethod
 SiteWaterMainsSchedule
 SiteWaterMainsAnnualAverageOutdoorAirTemperature
 SiteWaterMainsMaxDiffInMonthlyAverageOutdoorTemperature
 DesignAnnualPrecipitation
 NominalAnnualPrecipitation
 PrecipitationSchedule
 IrrigationType
 MaxIrrigationRate
 IrrigationSchedule
 CO2Concentration
 CO2Schedule
 ContaminantConcentration
 ContaminantSchedule
 TimeZoned
 UseDaylightSaving
 WinterFrom
 WinterTo
 SummerFrom
 SummerTo
 HourlyWeatherData
 DayOfWeekForStartDay
 UseWeatherFileSnowAndRain
 WeatherWinterDB996
 WinterToaDB996
 WinterWSDB996
 WinterWDD996
 WeatherWinterDB99
 WinterToaDB99
 WinterWSDB99
 WinterWDD99
 HeatingAutosizeMethod
 HeatingSimD1
 HeatingSimM1
 HeatingSimD2
 HeatingSimM2
 HeatingPeriodSelection
 SkySolarModelIndicator
 SkyBeamSolarDaySchedule
 SkyDiffuseSolarDaySchedule
 SkyASHRAEClearSkyBeamTaub
 SkyASHRAEClearSkyDiffuseTaud
 SkyClearness
 SummerToaLag
 SummerTwbLag
 SummerDBTempRangeModifierType
 SummerDBTempRangeModifierSchedule
 SummerDBTempProfileSchedule
 SummerDBHumidityConditionType
 SummerDBHumidityConditionSchedule
 SummerWSDB
 SummerWDD9
 CoolingAutosizeMethod
 CoolingSimD1
 CoolingSimM1
 CoolingSimD2
 CoolingSimM2
 CoolingPeriodSelection
 SummerTemperatureDefinition
 SummerDB04
 SummerToaMaxDB04

SummerTwbMaxDB04
 SummerToaMinDB04
 SummerDB1
 SummerToaMaxDB1
 SummerTwbMaxDB1
 SummerToaMinDB1
 SummerDB2
 SummerToaMaxDB2
 SummerTwbMaxDB2
 SummerToaMinDB2
 SummerWB04
 SummerToaMaxWB04
 SummerTwbMaxWB04
 SummerToaMinWB04
 SummerWB1
 SummerToaMaxWB1
 SummerTwbMaxWB1
 SummerToaMinWB1
 SummerWB2
 SummerToaMaxWB2
 SummerTwbMaxWB2
 SummerToaMinWB2
 SummerDP04
 SummerToaMaxDP04
 SummerTwbMaxDP04
 SummerToaMinDP04
 SummerDP1
 SummerToaMaxDP1
 SummerTwbMaxDP1
 SummerToaMinDP1
 SummerDP2
 SummerToaMaxDP2
 SummerTwbMaxDP2
 SummerToaMinDP2
 DesDBCoincWB04
 DryBulb04Jan
 DryBulb04Feb
 DryBulb04Mar
 DryBulb04Apr
 DryBulb04May
 DryBulb04Jun
 DryBulb04Jul
 DryBulb04Aug
 DryBulb04Sep
 DryBulb04Oct
 DryBulb04Nov
 DryBulb04Dec
 WetBulbDryBulb04Jan
 WetBulbDryBulb04Feb
 WetBulbDryBulb04Mar
 WetBulbDryBulb04Apr
 WetBulbDryBulb04May
 WetBulbDryBulb04Jun
 WetBulbDryBulb04Jul
 WetBulbDryBulb04Aug
 WetBulbDryBulb04Sep
 WetBulbDryBulb04Oct
 WetBulbDryBulb04Nov
 WetBulbDryBulb04Dec
 DryMin04Jan
 DryMin04Feb
 DryMin04Mar
 DryMin04Apr
 DryMin04May
 DryMin04Jun
 DryMin04Jul
 DryMin04Aug
 DryMin04Sep
 DryMin04Oct
 DryMin04Nov
 DryMin04Dec
 DesDBCoincWB2
 DryBulb2Jan
 DryBulb2Feb
 DryBulb2Mar
 DryBulb2Apr
 DryBulb2May
 DryBulb2Jun
 DryBulb2Jul
 DryBulb2Aug
 DryBulb2Sep
 DryBulb2Oct
 DryBulb2Nov
 DryBulb2Dec
 WetBulbDryBulb2Jan
 WetBulbDryBulb2Feb
 WetBulbDryBulb2Mar
 WetBulbDryBulb2Apr
 WetBulbDryBulb2May
 WetBulbDryBulb2Jun
 WetBulbDryBulb2Jul
 WetBulbDryBulb2Aug
 WetBulbDryBulb2Sep
 WetBulbDryBulb2Oct
 WetBulbDryBulb2Nov
 WetBulbDryBulb2Dec
 DryMin2Jan
 DryMin2Feb
 DryMin2Mar
 DryMin2Apr
 DryMin2May
 DryMin2Jun
 DryMin2Jul
 DryMin2Aug
 DryMin2Sep
 DryMin2Oct
 DryMin2Nov

DryMin2Dec
 DesDBCoincWB5
 DryBulb5Jan
 DryBulb5Feb
 DryBulb5Mar
 DryBulb5Apr
 DryBulb5May
 DryBulb5Jun
 DryBulb5Jul
 DryBulb5Aug
 DryBulb5Sep
 DryBulb5Oct
 DryBulb5Nov
 DryBulb5Dec
 WetBulbDryBulb5Jan
 WetBulbDryBulb5Feb
 WetBulbDryBulb5Mar
 WetBulbDryBulb5Apr
 WetBulbDryBulb5May
 WetBulbDryBulb5Jun
 WetBulbDryBulb5Jul
 WetBulbDryBulb5Aug
 WetBulbDryBulb5Oct
 WetBulbDryBulb5Sep
 WetBulbDryBulb5Nov
 WetBulbDryBulb5Dec
 DryMin5Jan
 DryMin5Feb
 DryMin5Mar
 DryMin5Apr
 DryMin5May
 DryMin5Jun
 DryMin5Jul
 DryMin5Aug
 DryMin5Sep
 DryMin5Oct
 DryMin5Nov
 DryMin5Dec
 DesDBCoincWB10
 DryBulb10Jan
 DryBulb10Feb
 DryBulb10Mar
 DryBulb10Apr
 DryBulb10May
 DryBulb10Jun
 DryBulb10Jul
 DryBulb10Aug
 DryBulb10Sep
 DryBulb10Oct
 DryBulb10Nov
 DryBulb10Dec
 WetBulbDryBulb10Jan
 WetBulbDryBulb10Feb
 WetBulbDryBulb10Mar
 WetBulbDryBulb10Apr
 WetBulbDryBulb10May
 WetBulbDryBulb10Jun
 WetBulbDryBulb10Jul
 WetBulbDryBulb10Aug
 WetBulbDryBulb10Sep
 WetBulbDryBulb10Oct
 WetBulbDryBulb10Nov
 WetBulbDryBulb10Dec
 DryMin10Jan
 DryMin10Feb
 DryMin10Mar
 DryMin10Apr
 DryMin10May
 DryMin10Jun
 DryMin10Jul
 DryMin10Aug
 DryMin10Sep
 DryMin10Oct
 DryMin10Nov
 DryMin10Dec
 DesWBCoincDB04
 WetBulb04Jan
 WetBulb04Feb
 WetBulb04Mar
 WetBulb04Apr
 WetBulb04May
 WetBulb04Jun
 WetBulb04Jul
 WetBulb04Aug
 WetBulb04Sep
 WetBulb04Oct
 WetBulb04Nov
 WetBulb04Dec
 DryBulbWetBulb04Jan
 DryBulbWetBulb04Feb
 DryBulbWetBulb04Mar
 DryBulbWetBulb04Apr
 DryBulbWetBulb04May
 DryBulbWetBulb04Jun
 DryBulbWetBulb04Jul
 DryBulbWetBulb04Aug
 DryBulbWetBulb04Sep
 DryBulbWetBulb04Oct
 DryBulbWetBulb04Nov
 DryBulbWetBulb04Dec
 WetMin04Jan
 WetMin04Feb
 WetMin04Mar
 WetMin04Apr
 WetMin04May
 WetMin04Jun
 WetMin04Jul
 WetMin04Aug
 WetMin04Sep
 WetMin04Oct
 WetMin04Nov
 WetMin04Dec

WetMin04Aug	WetMin10Apr	CostModel
WetMin04Sep	WetMin10May	OPTCalcDescription
WetMin04Oct	WetMin10Jun	CFDCalcDescription
WetMin04Nov	WetMin10Jul	CFDSimulationTurbulenceModel
WetMin04Dec	WetMin10Aug	CFDSimulationState
DesWBCoincDB2	WetMin10Sep	CFDSimulationViscosityOption
WetBulb2Jan	WetMin10Oct	CFDSimulationMolecularViscosityMultiplier
WetBulb2Feb	WetMin10Nov	CFDSimulationTurbulentViscosityValue
WetBulb2Mar	WetMin10Dec	CFDSimulationDiscretisationScheme
WetBulb2Apr	RegionId	CFDSimulationOuterIterations
WetBulb2May	IncludeElectricity	CFDSimulationIsothermal
WetBulb2Jun	IncludeNaturalGas	CFDSimulationSurfaceHeatTransferOption
WetBulb2Jul	IncludeFuelOil1	CFDSimulationCeilingHTCoefficient
WetBulb2Aug	IncludeFuelOil2	CFDSimulationWallHTCoefficient
WetBulb2Sep	IncludeCoal	CFDSimulationFloorHTCoefficient
WetBulb2Oct	IncludeGasoline	CFDSimulationInitialUComponent
WetBulb2Nov	IncludePropane	CFDSimulationInitialVComponent
WetBulb2Dec	IncludeDiesel	CFDSimulationInitialWComponent
DryBulbWetBulb2Jan	IncludeOtherFuel1	CFDSimulationInitialTemperature
DryBulbWetBulb2Feb	IncludeOtherFuel2	CFDSimulationUComponentInnerIterations
DryBulbWetBulb2Mar	StdECMinimal	CFDSimulationUComponentFalseTimeStep
DryBulbWetBulb2Apr	StdECNormal	CFDSimulationUComponentRelaxationFactor
DryBulbWetBulb2May	StdECMandatory	CFDSimulationUComponentTerminationResidual
DryBulbWetBulb2Jun	StdECBest	CFDSimulationVComponentInnerIterations
DryBulbWetBulb2Jul	OccupancyHVACTiming	CFDSimulationVComponentFalseTimeStep
DryBulbWetBulb2Aug	OccupancyGainsTiming	CFDSimulationVComponentRelaxationFactor
DryBulbWetBulb2Sep	FloorCeilingRepresentation	CFDSimulationVComponentTerminationResidual
DryBulbWetBulb2Oct	RValueToOutside	CFDSimulationWComponentInnerIterations
DryBulbWetBulb2Nov	ExcludeSemiExteriorUnconditionedZones	CFDSimulationWComponentFalseTimeStep
DryBulbWetBulb2Dec	SSComfortVelocity	CFDSimulationWComponentRelaxationFactor
WetMin2Jan	AirflowUnits	CFDSimulationWComponentTerminationResidual
WetMin2Feb	OccupancyLatentGainMethod	CFDSimulationPressureInnerIterations
WetMin2Mar	InfiltrationDataLevel	CFDSimulationTemperatureInnerIterations
WetMin2Apr	GainsDataLevel	CFDSimulationTemperatureFalseTimeStep
WetMin2May	CostDataLevel	CFDSimulationTemperatureRelaxationFactor
WetMin2Jun	CFDDDataLevel	CFDSimulationTemperatureTerminationResidual
WetMin2Jul	OccupantDataLevel	CFDSimulationTurbulenceKEInnerIterations
WetMin2Aug	OccupantDataEnableRandomOverride	CFDSimulationTurbulenceKEFalseTimeStep
WetMin2Sep	OccupantDataEnableRandomOverrideValue	CFDSimulationTurbulenceKERelaxationFactor
WetMin2Oct	OccupantDataEnableShades	CFDSimulationTurbulenceKETerminationResidual
WetMin2Nov	OccupantDataEnableWindows	CFDSimulationTurbulenceDissipationRateInnerIterations
WetMin2Dec	OccupantDataEnableLights	CFDSimulationTurbulenceDissipationRateFalseTimeStep
DesWBCoincDB5	OccupantDataEnableGains	CFDSimulationTurbulenceDissipationRateRelaxationFactor
WetBulb5Jan	GeometryDataLevel	CFDSimulationTurbulenceDissipationRateTerminationResidual
WetBulb5Feb	NatVentDataLevel	CFDSimulationViscosityRelaxationFactor
WetBulb5Mar	ScopeDataLevel	CFDSimulationRadiationDiscretisationScheme
WetBulb5Apr	RTIntegrationBaie	CFDSimulationRadiationMaximumIterations
WetBulb5May	ConstructionDataLevel	CFDSimulationRadiationRelativeTolerance
WetBulb5Jun	MechVentDataLevel	CFDSimulationRadiationRelaxationFactor
WetBulb5Jul	ShadingDataLevel	CFDSimulationRadiationMaximumResidual
WetBulb5Aug	SimulationDataLevelDBS	CFDSimulationRadiationNumberOfDirections
WetBulb5Sep	SimulationDataLevel	CFDSimulationHumidityTransportDiscretisationScheme
WetBulb5Oct	HVACDataLevel	CFDSimulationHumidityTransportMaximumIterations
WetBulb5Nov	SimpleHVACAUTOsizeMethod	CFDSimulationHumidityTransportRelativeTolerance
WetBulb5Dec	SimpleHVACAux	CFDSimulationHumidityTransportRelaxationFactor
DryBulbWetBulb5Jan	SimpleHVACSpec	CFDSimulationHumidityTransportMaximumResidual
DryBulbWetBulb5Feb	SimpleHVACMechVent	CFDSimulationScalarTransportDiscretisationScheme
DryBulbWetBulb5Mar	DetailedHVACSetpointDefinitionMethod	CFDSimulationScalarTransportMaximumIterations
DryBulbWetBulb5Apr	CalcNatVentAirTightnessMethod	CFDSimulationScalarTransportRelativeTolerance
DryBulbWetBulb5May	TimingDataLevel	CFDSimulationScalarTransportRelaxationFactor
DryBulbWetBulb5Jun	SimReducedWeather	CFDSimulationScalarTransportMaximumResidual
DryBulbWetBulb5Jul	PlantAdequate	CFDSimulationContinuityTerminationResidual
DryBulbWetBulb5Aug	PlantSized	CFDSimulationXAxisVelocity
DryBulbWetBulb5Sep	BuildingRegs	CFDSimulationYAxisVelocity
DryBulbWetBulb5Oct	WinterDesign	CFDSimulationZAxisVelocity
DryBulbWetBulb5Nov	WinterDesignIntermittent	CFDSimulationTemperature
DryBulbWetBulb5Dec	NatVentUserDefined	CFDSimulationPressure
WetMin5Jan	NatVentCalculated	CFDSimulationMass
WetMin5Feb	SSControlRF	CFDSimulationTurbulenceKE
WetMin5Mar	HLTemperatureControlType	CFDSimulationDissipationRateOfTurbulenceKE
WetMin5Apr	HGTemperatureControlType	CFDSimulationViscosity
WetMin5May	SSTemperatureControlType	CFDAuxiliaryCalculationIterations
WetMin5Jun	HGAirflowCalcMethod	CFDAuxiliaryCalculationTerminationError
WetMin5Jul	HLFast3TC	CFDAirChangeEffectiveness
WetMin5Aug	HGFast3TC	CFDSimulationBCFDKNumberOfCores
WetMin5Sep	SSFast3TC	CFDSimulationBCFDKConvergenceCriteria
WetMin5Oct	HLUseHeatingRadiantFraction	CFDSimulationBCFDKWritePeriod
WetMin5Nov	ExcludeNatVentHL	CFDSimulationBCFDKTurbulenceModel
WetMin5Dec	ExcludeMechVentHL	CFDSimulationInitialPressure
DesWBCoincDB10	ExcludeNatVentHG	CFDSimulationInitialTurbulenceKE
WetBulb10Jan	ExcludeMechVentHG	CFDSimulationInitialTurbulenceDissipationRate
WetBulb10Feb	ExcludeHeatRecoveryHG	CFDSimulationInitialSpecificDissipationRate
WetBulb10Mar	SSComfortRF	CFDSimulationContinuityEquationMaximumIterations
WetBulb10Apr	OCUDIFFSS	CFDSimulationContinuityEquationRelativeTolerance
WetBulb10May	OCUScheduleDef	CFDSimulationContinuityEquationRelaxationFactor
WetBulb10Jun	ThermalOcuOps	CFDSimulationContinuityEquationMaximumResidual
WetBulb10Jul	ThermalScheduleDef	CFDSimulationMomentumEquationDiscretisationScheme
WetBulb10Aug	ForceUpdateShadingSS	CFDSimulationMomentumEquationMaximumIterations
WetBulb10Sep	ForceUpdateShadingHG	CFDSimulationMomentumEquationRelativeTolerance
WetBulb10Oct	IncludeAllBuildingsSS	CFDSimulationMomentumEquationRelaxationFactor
WetBulb10Nov	IncludeAllBuildingsHG	CFDSimulationMomentumEquationMaximumResidual
WetBulb10Dec	ModelReflectionsHG	CFDSimulationEnergyEquationDiscretisationScheme
DryBulbWetBulb10Jan	ModelReflectionsSS	CFDSimulationEnergyEquationMaximumIterations
DryBulbWetBulb10Feb	AutosizeSS	CFDSimulationEnergyEquationRelativeTolerance
DryBulbWetBulb10Mar	GainDefinitionLevel	CFDSimulationEnergyEquationRelaxationFactor
DryBulbWetBulb10Apr	LoadsMethod	CFDSimulationEnergyEquationMaximumResidual
DryBulbWetBulb10May	PlantSizing	CFDSimulationAgeOfAirEquationDiscretisationScheme
DryBulbWetBulb10Jun	HGControlRF	CFDSimulationAgeOfAirEquationMaximumIterations
DryBulbWetBulb10Jul	HGControlRF	CFDSimulationAgeOfAirEquationRelativeTolerance
DryBulbWetBulb10Aug	HLControlRF	CFDSimulationAgeOfAirEquationRelaxationFactor
DryBulbWetBulb10Sep	HLComfortRF	CFDSimulationAgeOfAirEquationMaximumResidual
DryBulbWetBulb10Oct	SSCalcDescription	4281
DryBulbWetBulb10Nov	HLCalcDescription	CFDSimulationTurbulenceEquationMaximumIterations
DryBulbWetBulb10Dec	HGCalcDescription	CFDSimulationTurbulenceEquationRelativeTolerance
WetMin10Jan	ReportDescription	CFDSimulationTurbulenceEquationRelaxationFactor
WetMin10Feb	ReportType	CFDSimulationTurbulenceEquationMaximumResidual
WetMin10Mar	CostDecimalPlaces	

RunSimForSizingPeriod	SSEPTimestep	EndPointSnap
RunSimForHourlyWeather	SS3TCTimestep	DXFSnap
SimD1	SSDBSTimestep	OffsetSnap
SimM1	SaveBuildingTitle	OffsetSnapDistance
SimY1	Version	NearestPointSnap
SimD2	SSControlRFDDBS	XDrawingGuides
SimM2	SSComfortRFDDBS	YDrawingGuides
SimY2	OCUDISSDBS	HVACZoneAirDistributionUnitType
HeatGainD1	SimIModeDBS	HVACLoopTemplateType
HeatGainM1	HGSimIModeDBS	HVACZoneForcedAirUnitType
HeatGainM2	HGDBSTimestep	HVACZoneBaseboardType
HGStartMonth	HGComfortRFDDBS	BooleanOperation
HGEndMonth	HGControlRFDDBS	BooleanDifferenceToggle
RunSimForMultipleYears	HLSimIModeDBS	XAxisLock
SSYearsInSimulation	HLComfortRFDDBS	YAxisLock
MonthlyOutput	HLControlRFDDBS	ZAxisLock
DailyOutput	ZoneColourShadingMode	SwitchSnapDirection
HourlyOutput	AutomaticHVACNetworkBreaks	OutlineBlockConversionType
TimesteplyOutput	EnergyCodeld	QuantityMeasured
SurfacesOutput	RatingAssessmentType	TimeOfDay
SiteRecordSimStart	RatingAssessmentTypeIT	DayOfMonth
AllowCondConv	IEEConfirmConstructionsConform	MonthOfYear
AllowLighting	IEEKitchenPercElectric	SunpathDiagram
AllowOccupancy	IEERenDHW	SunpathDiagramScaleFactor
AllowProcess	IEERenDHWOther	SunpathDayOfMonth
AllowComputers	IEERenDHWFuel	RenderViewShadows
AllowCatering	IEERenCooling	SolarAltitude
AllowMiscellaneous	IEERenCoolingOther	SolarAzimuth
AllowEquipment	IEERenCoolingFuel	IncludeWindowFramesInShadowCreation
AllowLumped	IEERenHeating	FullSceneAntialiasingIndex
HLOverDesign	IEERenHeatingOther	FieldOfViewAngle
HLUseSteadyState	IEERenHeatingFuel	FullSceneAntialiasing
HLUseIntermittent	IEERenElectricity	TextureRenderMode
HGOverDesign	IEESimLighting	ShowNorthArrowInRenderedView
HGMaxCoolingTime	IEELighting	ShowGroundPlane
SingleZonePBIndex	IEESimElectricEquipment	EditDynamics
SingleZoneZoneIndex	IEEElectricEquipment	VisualisationDynamics
OccupancyScheduleScratch	IEESimDHW	DrawBuildingOpeningsInColour
HGDataAvailable	IEESimDHWElec	ShowWindows
HLDataAvailable	IEEDHW	DisplayZoneLabels
SSDataAvailable	IEEDHWElec	FitToIncludeDrawingFile
HGSimMethod	IEESimAux	ShowDrawingDownToSurfaceLevel
HLSimMethod	IEEAux	DisplayShadingDevicesInEditView
SSSimMethod	IEESimPumps	MeasureQuantity
EnergyRatingSimMethod	IEEPumps	BuildingBlockSnapMode
SSEPSolarDistribution	IEEAuxPercHeating	AssemblyName
SSEPSolutionAlgo	IEEAuxPercCooling	AssemblyCreationMode
SSAllowConstructionSolutionOverride	IEEAuxPercOther	pbEditUI
SSEPSpaceDiscretizationConstant	IEESimHeating	AssemblyDatumOffset
SSEPRelaxationFactor	IEESimHeatingElec	AssemblyXScaleFactor
SSEPInsideFaceSurfaceTemperatureConvergenceCriteria	IEEHeating	AssemblyYScaleFactor
SSEPDifferenceScheme	IEEHeatingElec	AssemblyZScaleFactor
SSEPTemperatureConv	IEESimCooling	AssemblyWidth
SSEPLoadsConv	IEESimCoolingElec	AssemblyHeight
AirflowNetworkMaxIterations	IEECooling	AssemblyDepth
AirflowNetworkAbsoluteConvTol	IEECoolingElec	MovieOutput
AirflowNetworkRelativeConvTol	IEESimExtLighting	MovieType
SSEPShadowingCalcMethod	IEEExtLightingPower	MovieOrbitIncrement
SSEPShadowingInterval	IEEExtLightingHours	MovieShadingTimeIncrement
SSEPShadowingSkyDiffuseModelingAlgorithm	IEEOtherElec	GenGeometryData
SSEPMaxWarmupDays	IEEFactorDeForma	GenPercGlazing
SSEPMinWarmupDays	IEEOvrerrideFactorDeForma	RepresentativeFloorArea
SSEPCalculateExternalSolarShading	RTModeSimu	GenWallToFloorAreaRatio
SSEPDesignAveragingWindow	RTBatEnCours	GenPartitionToFloorAreaRatio
LumpZonesBySelection	RTTousBat	GenGroundToFloorAreaRatio
LumpZonesByActivity	RTOptionSensibilite	GenRoofToFloorAreaRatio
LumpSBEMSurfaces	RTGenererFicheXML	GenCeilingToFloorAreaRatio
IEEPonderado	RTCalculAvecMP	GenFloorToFloorAreaRatio
LumpSimilarSBEMPartitions	RTDistOrthogonaleMaxMP	GenFloorArea
LumpSBEMZones	RTDistLateraleMaxMP	GenVolume
LumpZonesWhenConnected	RTRatioMinOmbrageMP	ZoneLumpedVolume
LumpElements	RTAngleMinMasqAzimutal	AnyNonDefaultData
LumpOpenings	RTLancerFichExterne	ZoneHandle
LumpCracks	RTFichierALancer	SiteStartUpOptions
LumpGains	SBEMProduceEPCXML	LockPassword
WindowMultiplier	SBEMXMLInputData	BuildingAddress0
Triangulate	XMLExportFolder	BuildingAddress1
ScheduledMixingFlowPerSqMetre	CodeAssessmentType	BuildingAddress2
ScheduledMixingOn	SuggestECMs	BuildingAddress3
ModelHoles	CalculateECMs	BuildingCity
WindowHoleDischargeCoefficient	LineType	BuildingCounty
FilterElementArea	PartitionOffset	BuildingPostcode
AllowPitchedRoof	InternalPartitionType	YearPlannedConstructionCompletionLEED
RoofWallTransitionAngle	BlockWallThickness	AnalystDetailsHead
FlatPitchedRoofTransitionAngle	BlockRoofThickness	AnalystName
FloorWallTransitionAngle	AutoCalculatePitchedRoofThickness	AnalystTelephone
LumpElementsNumberOrientations	UseGetRoofOverhang	AnalystAddress
HGEPSolarDistribution	PartitionThickness	AnalystCity
HGEPSolutionAlgo	IncrementSnap	AnalystPostcode
HGEPDifferenceScheme	ParallelSnap	SBEMWelshLanguage
HGEPSpaceDiscretizationConstant	NormalSnap	SBEMAlsoDoRegs
HGEPRelaxationFactor	SnapIncrement	SBEMRunEWEnergyRating
HGEPIinsideFaceSurfaceTemperatureConvergenceCriteria	SnapIncrementIP	SBEMEWEnergyRating
HGEPTemperatureConv	SnapToLowerAdjacentPerimeters	BuildingLocationDescription
HGEPLoadsConv	SnapToSurfaceAdjacentPerimeters	BuildingUPRN
HLEPTemperatureConv	ObjectRotationAngle	BuildingMPRN
HLEPLoadsConv	ArcSweepAngle	BuildingInspectionDate
YearJan1	ArcNumberSegments	BuildingConstructionDate
IsBookmark	AutoCompleteBlock	BuildingConservation
Handle	AutomaticInterBlockPartitionConnection	RTNumPermisConstruire
EditPerimeterAttributes	ProtractorIncrAngle	RTDateDepotPermisConstruire
AutomaticVoidInclusion	UseProtractor	RTAvancement
HGEnergyPlusData	PerimeterDrawMode	RTTypeTravaux
HLEnergyPlusData	OpeningDrawMode	RTModeProdChauffage
SEEnergyPlusData	CircleNumberSegments	NCMMMaxStores
HLReport	KeepDefaultOpenings	IsShellAndCore
HGReport	MidPointSnap	GasAvailable

GasAvailableS63
 RenewableMain
 IsModular
 MoreThan2Years
 MoreThan70Percent
 NoTERAvail
 DateOfSubAssemblies
 DateOfSubAssemblies2013
 DistressPurchase
 TransactionType
 AnalystNumber
 EAActivationCode
 UseProxyServer
 ProxyServerURL
 ProxyServerPort
 ProxyServerPassword
 ProxyServerUsername
 AnalystAccreditedScheme
 AnalystQualifications
 NOSLevel
 RelatedPartyDisclosureFromList
 AnalystEmployerName
 AnalystEmployerAddress
 AnalystEmployerTradingNumber
 AnalystInsuranceHead
 AnalystInsurerCompany
 AnalystInsurancePolicyNo
 AnalystInsuranceEffectiveDate
 AnalystInsuranceExpiryDate
 AnalystInsurancePILimit
 OwnerDetailsHead
 OwnerName
 OwnerTelephone
 OwnerAddress
 OwnerCity
 OwnerPostcode
 RTMOENom
 RTMOETelephone
 RTMOEAdresse
 RTMOEVille
 RTMOECodePostal
 RTBurCtrlNom
 RTBurCtrlTelephone
 RTBurCtrlAdresse
 RTBurCtrlVille
 RTBurCtrlCodePostal
 RTAnnexeV21Operation
 RTAnnexeV21OperationStatut
 RTAnnexeV22Systemes
 RTAnnexeV22SystemesStatut
 RTAnnexeV23ReseauChFr
 RTAnnexeV23ReseauChFrStatut
 Notes
 CFDWindVelocity
 CFDWindDirection
 CFDWindExposureIndex
 CFDSiteDomainLengthFactor
 CFDSiteDomainWidthFactor
 CFDSiteDomainHeightFactor
 CFDResults
 GroupBC
 AnalysisType
 CommandLine
 ProgramPath
 DaylightingMethod
 SBEMBRSTAGE
 SBEMBERSTAGE
 SBEMBR6BTYP
 SBEMScotlandRatingImprov
 SBEMSEPCCase
 SBEMBR6ACCREDCONST
 SBEMBR6NVCREDIT
 SBEMDataReflectionReports
 SBEMELECPOWERFACTOR
 SBEMDistrictHeatingIsDefault
 SBEMDistrictHeatingDefaultCO2Factor
 SBEMDistrictHeatingIsUser
 SBEMDistrictHeatingUserCO2Factor
 SBEMDistrictHeatingPEIsDefault
 SBEMDistrictHeatingDefaultPEFactor
 SBEMDistrictHeatingPEIsUser
 SBEMDistrictHeatingUserPEFactor
 SSEPSshadowOverlap
 HGEPSshadowOverlap
 SSEPPolygonClippingAlgorithm
 HGEPPolygonClippingAlgorithm
 AutomaticBlockZoning
 FollowPathSeparateBlocks
 SSIDFIncludeFile1
 SSIDFIncludeFilename1
 SSIDFIncludeFile2
 SSIDFIncludeFilename2
 FlatCBSselectionMethod
 FlatCBFaction2LargestAreas
 ComponentFlatExtrusionThickness
 SSEPAutosizingFactorHeating
 SSEPAutosizingFactorCooling
 HGEPCompactHVACType
 HGDesignOffCoilTemp
 HGDesignSupplyAirHumidityRatio
 HLModelInterzoneToSameZoneAsAdiabatic
 HGModelInterzoneToSameZoneAsAdiabatic
 SSModelInterzoneToSameZoneAsAdiabatic
 HGSupplyAirMinFlowFraction
 WaterOutletFlowRateData
 HLDateFormatFlag
 HGDateFormatFlag

SSDateFormatFlag
 WinterDesignDayGeneral
 WinterDesignDayOccupancy
 WinterDesignDayLighting
 WinterDesignDayEquipment
 WinterDesignDayHeating
 WinterDesignDayCooling
 WinterDesignDayHVAC
 WinterDesignDayNatVent
 WinterDesignDayDHW
 WinterDesignDay
 SummerDesignDay
 SSCheckConvex
 HGCheckConvex
 CFDViewType
 CFDDisplayShowKey
 DrawCFDGrid
 BCFDKMonitorPointDisplay
 BCFDKStreamlineSeedPointDisplay
 CFDDisplayMaximumVectorLength
 CFDDisplayVectorScaleFactor
 CFDDisplayMinimumVelocity
 CFDDisplayMaximumVelocity
 CFDDisplayVelocityContourBandValue1
 CFDDisplayVelocityContourBandValue2
 CFDDisplayVelocityContourBandValue3
 CFDDisplayVelocityContourBandValue4
 CFDDisplayVelocityContourBandValue5
 CFDDisplayVelocityContourBandValue6
 CFDDisplayVelocityContourBandValue7
 CFDDisplayVelocityContourBandValue8
 CFDDisplayVelocityContourBandValue9
 CFDDisplayVelocityContourBandValue10
 CFDDisplayVelocityContourBandValue11
 CFDDisplayVelocityContourBandValue12
 CFDDisplayVelocityContourBandActive1
 CFDDisplayVelocityContourBandActive2
 CFDDisplayVelocityContourBandActive3
 CFDDisplayVelocityContourBandActive4
 CFDDisplayVelocityContourBandActive5
 CFDDisplayVelocityContourBandActive6
 CFDDisplayVelocityContourBandActive7
 CFDDisplayVelocityContourBandActive8
 CFDDisplayVelocityContourBandActive9
 CFDDisplayVelocityContourBandActive10
 CFDDisplayVelocityContourBandActive11
 CFDDisplayVelocityContourBandActive12
 CFDDisplayMinimumTemperature
 CFDDisplayMaximumTemperature
 CFDDisplayTemperatureContourBandValue1
 CFDDisplayTemperatureContourBandValue2
 CFDDisplayTemperatureContourBandValue3
 CFDDisplayTemperatureContourBandValue4
 CFDDisplayTemperatureContourBandValue5
 CFDDisplayTemperatureContourBandValue6
 CFDDisplayTemperatureContourBandValue7
 CFDDisplayTemperatureContourBandValue8
 CFDDisplayTemperatureContourBandValue9
 CFDDisplayTemperatureContourBandValue10
 CFDDisplayTemperatureContourBandValue11
 CFDDisplayTemperatureContourBandValue12
 CFDDisplayTemperatureContourBandActive1
 CFDDisplayTemperatureContourBandActive2
 CFDDisplayTemperatureContourBandActive3
 CFDDisplayTemperatureContourBandActive4
 CFDDisplayTemperatureContourBandActive5
 CFDDisplayTemperatureContourBandActive6
 CFDDisplayTemperatureContourBandActive7
 CFDDisplayTemperatureContourBandActive8
 CFDDisplayTemperatureContourBandActive9
 CFDDisplayTemperatureContourBandActive10
 CFDDisplayTemperatureContourBandActive11
 CFDDisplayTemperatureContourBandActive12
 CFDDisplayMinimumPressure
 CFDDisplayMaximumPressure
 CFDDisplayPressureContourBandValue1
 CFDDisplayPressureContourBandValue2
 CFDDisplayPressureContourBandValue3
 CFDDisplayPressureContourBandValue4
 CFDDisplayPressureContourBandValue5
 CFDDisplayPressureContourBandValue6
 CFDDisplayPressureContourBandValue7
 CFDDisplayPressureContourBandValue8
 CFDDisplayPressureContourBandValue9
 CFDDisplayPressureContourBandValue10
 CFDDisplayPressureContourBandValue11
 CFDDisplayPressureContourBandValue12
 CFDDisplayPressureContourBandActive1
 CFDDisplayPressureContourBandActive2
 CFDDisplayPressureContourBandActive3
 CFDDisplayPressureContourBandActive4
 CFDDisplayPressureContourBandActive5
 CFDDisplayPressureContourBandActive6
 CFDDisplayPressureContourBandActive7
 CFDDisplayPressureContourBandActive8
 CFDDisplayPressureContourBandActive9
 CFDDisplayPressureContourBandActive10
 CFDDisplayPressureContourBandActive11
 CFDDisplayPressureContourBandActive12
 CFDDisplayMinimumH2OMassFraction
 CFDDisplayMaximumH2OMassFraction
 CFDDisplayH2OMassFractionContourBandValue1
 CFDDisplayH2OMassFractionContourBandValue2
 CFDDisplayH2OMassFractionContourBandValue3
 CFDDisplayH2OMassFractionContourBandValue4
 CFDDisplayH2OMassFractionContourBandValue5
 CFDDisplayH2OMassFractionContourBandValue6

CFDDisplayH2OMassFractionContourBandValue7
 CFDDisplayH2OMassFractionContourBandValue8
 CFDDisplayH2OMassFractionContourBandValue9
 CFDDisplayH2OMassFractionContourBandValue10
 CFDDisplayH2OMassFractionContourBandValue11
 CFDDisplayH2OMassFractionContourBandValue12
 CFDDisplayH2OMassFractionContourBandActive1
 CFDDisplayH2OMassFractionContourBandActive2
 CFDDisplayH2OMassFractionContourBandActive3
 CFDDisplayH2OMassFractionContourBandActive4
 CFDDisplayH2OMassFractionContourBandActive5
 CFDDisplayH2OMassFractionContourBandActive6
 CFDDisplayH2OMassFractionContourBandActive7
 CFDDisplayH2OMassFractionContourBandActive8
 CFDDisplayH2OMassFractionContourBandActive9
 CFDDisplayH2OMassFractionContourBandActive10
 CFDDisplayH2OMassFractionContourBandActive11
 CFDDisplayH2OMassFractionContourBandActive12
 CFDDisplayMinimumCO2MassFraction
 CFDDisplayMaximumCO2MassFraction
 CFDDisplayCO2MassFractionContourBandValue1
 CFDDisplayCO2MassFractionContourBandValue2
 CFDDisplayCO2MassFractionContourBandValue3
 CFDDisplayCO2MassFractionContourBandValue4
 CFDDisplayCO2MassFractionContourBandValue5
 CFDDisplayCO2MassFractionContourBandValue6
 CFDDisplayCO2MassFractionContourBandValue7
 CFDDisplayCO2MassFractionContourBandValue8
 CFDDisplayCO2MassFractionContourBandValue9
 CFDDisplayCO2MassFractionContourBandValue10
 CFDDisplayCO2MassFractionContourBandValue11
 CFDDisplayCO2MassFractionContourBandValue12
 CFDDisplayCO2MassFractionContourBandActive1
 CFDDisplayCO2MassFractionContourBandActive2
 CFDDisplayCO2MassFractionContourBandActive3
 CFDDisplayCO2MassFractionContourBandActive4
 CFDDisplayCO2MassFractionContourBandActive5
 CFDDisplayCO2MassFractionContourBandActive6
 CFDDisplayCO2MassFractionContourBandActive7
 CFDDisplayCO2MassFractionContourBandActive8
 CFDDisplayCO2MassFractionContourBandActive9
 CFDDisplayCO2MassFractionContourBandActive10
 CFDDisplayCO2MassFractionContourBandActive11
 CFDDisplayCO2MassFractionContourBandActive12
 CFDDisplayMinimumAgeOfAir
 CFDDisplayMaximumAgeOfAir
 CFDDisplayAgeOfAirContourBandValue1
 CFDDisplayAgeOfAirContourBandValue2
 CFDDisplayAgeOfAirContourBandValue3
 CFDDisplayAgeOfAirContourBandValue4
 CFDDisplayAgeOfAirContourBandValue5
 CFDDisplayAgeOfAirContourBandValue6
 CFDDisplayAgeOfAirContourBandValue7
 CFDDisplayAgeOfAirContourBandValue8
 CFDDisplayAgeOfAirContourBandValue9
 CFDDisplayAgeOfAirContourBandValue10
 CFDDisplayAgeOfAirContourBandValue11
 CFDDisplayAgeOfAirContourBandValue12
 CFDDisplayAgeOfAirContourBandActive1
 CFDDisplayAgeOfAirContourBandActive2
 CFDDisplayAgeOfAirContourBandActive3
 CFDDisplayAgeOfAirContourBandActive4
 CFDDisplayAgeOfAirContourBandActive5
 CFDDisplayAgeOfAirContourBandActive6
 CFDDisplayAgeOfAirContourBandActive7
 CFDDisplayAgeOfAirContourBandActive8
 CFDDisplayAgeOfAirContourBandActive9
 CFDDisplayAgeOfAirContourBandActive10
 CFDDisplayAgeOfAirContourBandActive11
 CFDDisplayAgeOfAirContourBandActive12
 CFDDisplayMinimumPPD
 CFDDisplayMaximumPPD
 CFDDisplayPPDContourBandValue1
 CFDDisplayPPDContourBandValue2
 CFDDisplayPPDContourBandValue3
 CFDDisplayPPDContourBandValue4
 CFDDisplayPPDContourBandValue5
 CFDDisplayPPDContourBandValue6
 CFDDisplayPPDContourBandValue7
 CFDDisplayPPDContourBandValue8
 CFDDisplayPPDContourBandValue9
 CFDDisplayPPDContourBandValue10
 CFDDisplayPPDContourBandValue11
 CFDDisplayPPDContourBandValue12
 CFDDisplayPPDContourBandActive1
 CFDDisplayPPDContourBandActive2
 CFDDisplayPPDContourBandActive3
 CFDDisplayPPDContourBandActive4
 CFDDisplayPPDContourBandActive5
 CFDDisplayPPDContourBandActive6
 CFDDisplayPPDContourBandActive7
 CFDDisplayPPDContourBandActive8
 CFDDisplayPPDContourBandActive9
 CFDDisplayPPDContourBandActive10
 CFDDisplayPPDContourBandActive11
 CFDDisplayPPDContourBandActive12
 CFDDisplayMinimumPMV
 CFDDisplayMaximumPMV
 CFDDisplayPMVContourBandValue1
 CFDDisplayPMVContourBandValue2
 CFDDisplayPMVContourBandValue3
 CFDDisplayPMVContourBandValue4
 CFDDisplayPMVContourBandValue5
 CFDDisplayPMVContourBandValue6
 CFDDisplayPMVContourBandValue7
 CFDDisplayPMVContourBandValue8
 CFDDisplayPMVContourBandValue9

CFDDisplayPMVContourBandValue10
 CFDDisplayPMVContourBandValue11
 CFDDisplayPMVContourBandValue12
 CFDDisplayPMVContourBandActive1
 CFDDisplayPMVContourBandActive2
 CFDDisplayPMVContourBandActive3
 CFDDisplayPMVContourBandActive4
 CFDDisplayPMVContourBandActive5
 CFDDisplayPMVContourBandActive6
 CFDDisplayPMVContourBandActive7
 CFDDisplayPMVContourBandActive8
 CFDDisplayPMVContourBandActive9
 CFDDisplayPMVContourBandActive10
 CFDDisplayPMVContourBandActive11
 CFDDisplayPMVContourBandActive12
 CFDDisplayMinimumMeanRadiantTemperature
 CFDDisplayMaximumMeanRadiantTemperature
 CFDDisplayMeanRadiantTemperatureContourBandValue1
 CFDDisplayMeanRadiantTemperatureContourBandValue2
 CFDDisplayMeanRadiantTemperatureContourBandValue3
 CFDDisplayMeanRadiantTemperatureContourBandValue4
 CFDDisplayMeanRadiantTemperatureContourBandValue5
 CFDDisplayMeanRadiantTemperatureContourBandValue6
 CFDDisplayMeanRadiantTemperatureContourBandValue7
 CFDDisplayMeanRadiantTemperatureContourBandValue8
 CFDDisplayMeanRadiantTemperatureContourBandValue9
 CFDDisplayMeanRadiantTemperatureContourBandValue10
 CFDDisplayMeanRadiantTemperatureContourBandValue11
 CFDDisplayMeanRadiantTemperatureContourBandValue12
 CFDDisplayMeanRadiantTemperatureContourBandActive1
 CFDDisplayMeanRadiantTemperatureContourBandActive2
 CFDDisplayMeanRadiantTemperatureContourBandActive3
 CFDDisplayMeanRadiantTemperatureContourBandActive4
 CFDDisplayMeanRadiantTemperatureContourBandActive5
 CFDDisplayMeanRadiantTemperatureContourBandActive6
 CFDDisplayMeanRadiantTemperatureContourBandActive7
 CFDDisplayMeanRadiantTemperatureContourBandActive8
 CFDDisplayMeanRadiantTemperatureContourBandActive9
 CFDDisplayMeanRadiantTemperatureContourBandActive10
 CFDDisplayMeanRadiantTemperatureContourBandActive11
 CFDDisplayMeanRadiantTemperatureContourBandActive12
 CFDDisplayMinimumOperativeTemperature
 CFDDisplayMaximumOperativeTemperature
 CFDDisplayOperativeTemperatureContourBandValue1
 CFDDisplayOperativeTemperatureContourBandValue2
 CFDDisplayOperativeTemperatureContourBandValue3
 CFDDisplayOperativeTemperatureContourBandValue4
 CFDDisplayOperativeTemperatureContourBandValue5
 CFDDisplayOperativeTemperatureContourBandValue6
 CFDDisplayOperativeTemperatureContourBandValue7
 CFDDisplayOperativeTemperatureContourBandValue8
 CFDDisplayOperativeTemperatureContourBandValue9
 CFDDisplayOperativeTemperatureContourBandValue10
 CFDDisplayOperativeTemperatureContourBandValue11
 CFDDisplayOperativeTemperatureContourBandValue12
 CFDDisplayOperativeTemperatureContourBandActive1
 CFDDisplayOperativeTemperatureContourBandActive2
 CFDDisplayOperativeTemperatureContourBandActive3
 CFDDisplayOperativeTemperatureContourBandActive4
 CFDDisplayOperativeTemperatureContourBandActive5
 CFDDisplayOperativeTemperatureContourBandActive6
 CFDDisplayOperativeTemperatureContourBandActive7
 CFDDisplayOperativeTemperatureContourBandActive8
 CFDDisplayOperativeTemperatureContourBandActive9
 CFDDisplayOperativeTemperatureContourBandActive10
 CFDDisplayOperativeTemperatureContourBandActive11
 CFDDisplayOperativeTemperatureContourBandActive12
 CFDDisplayVariableVelocityVectors
 CFDDisplayVariableVelocityContours
 CFDDisplayVariableTemperatureContours
 CFDDisplayVariablePressureContours
 CFDDisplayVariableH2OMassFractionContours
 CFDDisplayVariableCO2MassFractionContours
 CFDDisplayVariableAgeOfAirContours
 CFDDisplayVariablePMVContours
 CFDDisplayVariablePPDContours
 CFDDisplayVariableComfortContours
 CFDDisplayVariableMeanRadiantTemperatureContours
 CFDDisplayVariableOperativeTemperatureContours
 CFDDisplayVariableFilledVelocityContours
 CFDDisplayVariableFilledTemperatureContours
 CFDDisplayVariableFilledPressureContours
 CFDDisplayVariableFilledH2OMassFractionContours
 CFDDisplayVariableFilledCO2MassFractionContours
 CFDDisplayVariableFilledAgeOfAirContours
 CFDDisplayVariableFilledPMVContours
 CFDDisplayVariableFilledPPDContours
 CFDDisplayVariableFilledComfortContours
 CFDDisplayVariableFilledMeanRadiantTemperatureContours
 CFDDisplayVariableFilledOperativeTemperatureContours
 CFDDisplay3DContourNone
 CFDDisplay3DContourVelocity
 CFDDisplay3DContourTemperature
 CFDDisplay3DContourPressure
 CFDDisplay3DContourH2OMassFraction
 CFDDisplay3DContourCO2MassFraction
 CFDDisplay3DContourAgeOfAir
 CFDDisplay3DContourPMV
 CFDDisplay3DContourPPD
 CFDDisplay3DContourComfort
 CFDDisplay3DContourMeanRadiantTemperature
 CFDDisplay3DContourOperativeTemperature
 CFDGridRegionOperation
 CFDGridRegionStartCoordinate
 CFDGridRegionEndCoordinate
 CFDGridRegionNewCoordinate
 CFDGridRegionSpacingType
 CFDGridRegionSpacingDimension
 CFDGridRegionSpacingPower
 CFDGridRegionNumberOfDivisions
 CFDGridRegionLocked
 CFDMonitorCellIncluded
 CFDMonitorCellName
 CFDMonitorCellXCoordinate
 CFDMonitorCellYCoordinate
 CFDMonitorCellZCoordinate
 CFDDisplayStreamlineMinimumStepLength
 CFDDisplayStreamlineMaximumStepLength
 CFDDisplayStreamlineInitialStepLength
 CFDDisplayStreamlineMaximumLength
 CFDDisplayStreamlineMaximumError
 CFDDisplayStreamlineTerminalVelocity
 CFDDisplayStreamlineMaximumSteps
 CFDDisplayStreamlineIntegrationDirection
 CFDDisplayStreamlineIntegratorType
 CFDDisplayStreamlineInterpolatorType
 CFDDisplayStreamlineStepUnit
 CFDMonitorCellOperation
 CFDSimulationMonitorCellName
 CFDSimulationMonitorCellIndex
 CFDSimulationMonitorCellVariable
 CFDSimulationResultsCurrentIndex
 CFDSimulationResultsSetOption
 CFDExistingResultsSetOption
 CFDExistingResultsSet
 CFDSimulationInternalDefaultGridSpacing
 CFDSimulationInternalGridLineMergeTolerance
 CFDSimulationExternalDefaultGridSpacing
 CFDSimulationExternalGridLineMergeTolerance
 CFDSimulationExternalGridScheme
 CFDSiteDomainDisplay
 CFDDisplayWireFramePartitions
 CFDDisplayWireFrameComponentBlocks
 CFDDisplayWireFrameFloorSlabs
 ShadingPlaneIndex
 ObjectKey
 PAYGStatus
 IDBImportStatus
 MergeCoPlanarGBXMLShadeSurfaces
 ImportGBXMLBlockMode
 ImportGBXMLThermalProperties
 ImportGBXMLShadeSurfaces
 GBXMLImportMode
 GBXMLImportPriority
 AdjacencySeparationTolerance
 AdjacencyAngularTolerance
 StandardComponentBlockAdjacencies
 3DCADView
 3DCADFileType
 3DCADFilename
 OptimisationEngine
 OPTelitism
 OPTMaxGenerations
 OPTConvergenceGens
 OPTMaxPopulationSize
 OPTInitialPopulationSize
 OPTFloatEncodingLength
 OPTListEncodingLength
 OPTCrossoverRate
 OPTIndividualMutationProbability
 OPTOverrideBitwiseMutationProbability
 OPTBitwiseMutationProbability
 OPTTournamentSize
 SATtrajectories
 SATtotalRuns
 JEAAlgorithm
 JEASampleSize
 JEALnitSampleOption
 SAMethod
 JEALnitPopulationSize
 JEAEvolvePopSize
 JEAMaxPopSize
 JEAMutationRate
 JEACrossoverRate
 JEATournamentSize
 JEAObjectiveBias
 JEAMaxEvaluations
 JEAMaxComputingTime
 JEAMaxWallTime
 JEAValueStrSamples
 ConstraintLowerMargin
 ConstraintUpperMargin
 RetainBatchResults
 RMCalcDescription
 RMTraceType
 RMParameters
 RMMMaxCores
 RMAmbientDivisions
 RMAmbientBounces
 RMAmbientAccuracy
 RMAmbientResolution
 RMAmbientSuperSamples
 RMRandomSampling
 RMOVERRIDEZoneLuxTarget
 RMOVERRIDEZoneOccupancy
 RMExcludeUnlitHours
 RMGroundGlow
 RMPrintLogFile
 RMRemoveWorkingFiles
 RMMinGridSize
 RMMMaxGridSize
 RMMMargin
 RMGroundPlaneExtension
 RMUseMarginAvDF
 RMUseMarginMinMaxDF
 RMSky
 RMSkyMethod
 RMPerezMethod
 RMEpsilon
 RMDelta
 RMDiffuseHorizontalIrradiance
 RMDirectNormalIrradiance
 RMDirectHorizontalIrradiance
 RMDirectNormalIlluminance
 RMDiffuseHorizontalIlluminance
 RMEpsilonPM
 RMDeltaPM
 RMDiffuseHorizontalIrradiancePM
 RMDirectNormalIrradiancePM
 RMDirectHorizontalIrradiancePM
 RMDirectNormalIlluminancePM
 RMDiffuseHorizontalIlluminancePM
 RMSkyIlluminance
 RMDAThreshold
 RMASELuxThreshold
 RMASEHrsThreshold
 RMUDIUpperLuxThreshold
 RMUDIUpperLuxThreshold
 RMBreemAvgThreshold
 RMBreemMinThreshold
 RMIncludeOtherBuildings
 RMIncludeBlindGeometry
 RMUseDynamicShading
 RICalcDescription
 RITraceType
 RIParameters
 RISky
 RMRadianceSimulationType
 RMDaysimSimulationType
 RMSkyPatches
 RMSunPatches
 RMRadianceReportTypeGeneral
 RMRadianceReportTypeOthers
 RMDaysimReportTypeGeneral
 RMDaysimReportTypeLeed
 RMDaysimReportTypeBreeam
 RMMarginDisplay
 RMOccupancySchedule
 RMColourPalette
 RMCeilValueSelection
 RMCeilValueSelectionDaysim
 RMCeilValueSelectionSDA
 RMScaleValueSource
 RMScaleValueSourceDaysim
 RMCeilValueColourMode
 RMAMP
 RMScaleSource
 RMScaleLow
 RMScaleHigh
 RMScalePctLow
 RMScalePctHigh
 RMScaleHrsLow
 RMScaleHrsHigh
 RMApplyScaleLowColour
 RMApplyScaleHighColour
 RMScaleLowColour
 RMScaleHighColour
 RMDFTThreshold
 RMLuxThreshold
 RMMaxLuxThreshold
 RMMinUniformityRatio
 RMMinUniformityRatioAtria
 RMMinPointDF
 RMMinPointDFAtria
 RMPercAreaDaylit
 RMDAPctThreshold
 RMUDIPTThreshold
 RMBreemAvgHrsThreshold
 RMBreemMinHrsThreshold
 RMBreemPassThreshold
 RMDaylightThresholdType
 RMLEedConstructionType
 RMSkyDay
 RMSkyMonth
 RMSkyHour
 HVACZoneLDeviceRadiantSurfaceList
 HVACZoneChilledCeilingRadiantSurfaceList
 HVACZoneHeatedFloorRadiantSurfaceList
 HVACZoneRadiantSurfaceRadiantSurfaceList
 CFDCOMFORTActivityMetabolicRate
 CFDCOMFORTClothingLevel
 CFDCOMFORTRelativeHumidity
 CFDCOMFORTMRTNumberOfSphereSegments
 RMApPLYZoneLabels
 RMCeilValueUserColour
 HVACCoolingCoilType
 HVACComponentType
 BuildingGUID
 SiteGUID
 EEUseDirectLodgement
 EESurveyorID
 EESurveyorPassword
 EEReferenceNumber
 BatchProcessing
 UseJobServer
 JobServerSplitSimulation
 UseJobServerOPT
 OverrideOPTJobs
 OPTJobs
 JobServerList
 JobServer

JobServerTag	NRMCostStairsE	NumberSmoothScaleIterations
ToleranceTimeHeatingSetpointNotMet	NRMCostFitting	Tolerance
ToleranceTimeCoolingSetpointNotMet	NRMCostFittingA	NumberSolveIterations
IDFDuplicateNameDelimiter	NRMCostSanitary	NumberRelaxIterations
IDFReplacementCharacter	NRMCostSanitaryA	NumberFeatureSnapIterations
IDFBlockZoneSeparator	NRMCostServeEquip	MaximumNonOrthogonalAngleActive
OptimisationObjectives	NRMCostServeEquipA	MaximumNonOrthogonalAngle
OptimisationOutputs	NRMCostDisposal	MaximumBoundarySkewnessActive
OptimisationConstraints	NRMCostDisposalA	MaximumBoundarySkewness
OptimisationVariables	NRMCostWater	MaximumInternalSkewnessActive
AnalystEmail	NRMCostWaterA	MaximumInternalSkewness
AnalystAccreditedSchemeWeb	NRMCostElectrical	MaximumConcaveAngleActive
XMLRestricted	NRMCostElectricalA	MaximumConcaveAngle
EPBDAsk1	NRMCostFuel	MinimumVolumeActive
EPBDAsk2	NRMCostFuelA	MinimumVolume
EPBDAskRenewable	NRMCostLift	MinimumTetrahedralQualityActive
EPBDAskCHP	NRMCostLifeE	MinimumTetrahedralQuality
EPBDAskDH	NRMCostFire	MinimumFaceAreaActive
EPBDAskHP	NRMCostFireA	MinimumFaceArea
DataOptionTemplateld	NRMCostCommunication	MinimumFaceTwistActive
GridSnap	NRMCostCommunicationA	MinimumFaceTwist
GridXSpacing	NRMCostSpecialist	MinimumDeterminant
GridYSpacing	NRMCostSpecialistA	MinimumFaceWeight
ClippingPlaneHeight	NRMCostBuilders	MinimumVolumeRatio
ClippingPlaneOffset	NRMCostBuildersA	ErrorReductionFactor
DisplayZoneInnerSurfaces	NRMCostCompleted	NoQ50InfCheck
DisplayFilledExternalWalls	NRMCostCompletedA	NumberOfFloors
DisplayZoneInnerVolumeFloors	NRMCostMinorDemo	AnyOpeningOutput
DisplayZoneInnerFloorPerimeters	NRMCostMinorDemoA	DataPlotType
DisplayGBXMLExternalWallSurface	NRMCostRepairs	InputDataPlotType
DisplayGBXMLInternalWallSurface	NRMCostRepairsA	OutputDataPlotType
DisplayGBXMLShadeSurface	NRMCostDampProof	DataPlotFieldIndexExternalSurfaces
DisplayGBXMLRoofSurface	NRMCostDampProofA	DataPlotFieldIndexInternalSurfaces
DisplayGBXMLAirSurface	NRMCostFacadeRetention	DataPlotFieldIndexZoneData
DisplayGBXMLUnderGroundWallSurface	NRMCostFacadeRetentionA	DataPlotFieldIndex
DisplayGBXMLSlabOnGradeSurface	NRMCostCleaning	DataPlotUnits
DisplayGBXMLRaisedFloorSurface	NRMCostCleaningA	DataPlotInterval
DisplayGBXMLUndergroundSlabSurface	NRMCostRenovation	DataPlotTableName
DisplayGBXMLInteriorFloorSurface	NRMCostRenovationA	DataPlotPeriod
DisplayGBXMLUndergroundCeilingSurface	NRMCostSitePrep	DataPlotMonth
DisplayGBXMLCeilingSurface	NRMCostSitePrepA	DataPlotDay
DisplayGBXMLOtherSurface	NRMCostRoadPath	DataPlotHour
ClippingPlaneOrientationSwitchingMode	NRMCostRoadPathA	DataPlotSubHour
ZoneVolumeCalculationMethod	NRMCostSoftLandscape	DataPlotRecordIndex
ZoneSurfaceAreaCalculationMethod	NRMCostSoftLandscapeA	DataPlotValue
ZoneFloorAreaCalculationMethod	NRMCostFence	DataPlotMinValue
ReflectionAxis	NRMCostFenceA	DataPlotMaxValue
JobTypeList	NRMCostEFixture	DataPlotNormalise
JobType	NRMCostEFixtureA	DataPlotNumberBands
SBEMYear	NRMCostEDrainage	DataPlotMinBandColour
IDFNameType	NRMCostEDrainageA	DataPlotMaxBandColour
IDFCompression	NRMCostEService	DataPlotRangeMethod
gbXMLConstructionName	NRMCostEServiceA	DataPlotName
gbXMLWindowTypeName	NRMCostMinorWork	DataPlotLock
ClipSurfaceOpenings	NRMCostMinorWorkA	DataPlotDecimalPlaces
SurfaceOpeningClipTolerance	NRMCostMCEmployer	DataPlotVectorScaleFactor
EMSON	NRMCostMCCostItem	DataPlotBodyOpacity
FMUON	NRMCostMCOverhead	DataPlotShowAirflowVectorLabels
EMSId	NRMCostMCProfit	DataPlotAirflowMagnitudeLabelSize
FMUIId	NRMCostConsultant	DataPlotConstructionKeyLabelSize
SurfaceSelectionMode	NRMCostMCPreConstruct	DataPlotShowOpeningApertures
AllowOpenManifoldBuildingBlocks	NRMCostMCDesign	DataPlotWindSpeed
GBXMLImportMode	NRMCostOtherCost	DataPlotWindDirection
CombineBlocksInStores	NRMCostDesignRisk	DataAnalysisComponentDisplay
SavedFrameOuterWidth	NRMCostConstructionRisk	ShowVisualisationBlockExternalFaceEdges
SavedFrameInnerWidth	NRMCostEmployerRisk	ShowVisualisationZoneInternalFaceEdges
DetailedHVACFreeRun	NRMCostOtherRisk	L5TotalInternalGains
MergeHVACSystem	NRMCostTenderInflation	SurroundingBlockDisplay
MergeAssemblies	NRMCostConstructInflation	SurroundingBlockZoneDisplay
MergeComponentBlocks	UNIFORMATC20	SurroundingBuildingZoneDisplay
OpeningMergeOption	UNIFORMATC20E	ShowVisualisationSurroundingSurfaces
MergeFilename	UNIFORMATD10	ShowVisualisationSurroundingZones
LCALifeYears	UNIFORMATD10E	ShowVisualisationSurroundingBlocks
GlazingEmbodiedCO2KgPerKg	UNIFORMATD20	DisplayFilledZoneWalls
GlazingCoatingEmbodiedCO2KgPerM2	UNIFORMATD20A	Active
DetailedHVACTemplateld	UNIFORMATD40	LEED2009Cat
L5EnhVentRate	UNIFORMATD40A	LEED2009Application
L5EnhVentOffsetT1	UNIFORMATD50	LEEDv4Cat
L5EnhVentOffsetT2	UNIFORMATD50A	LEEDv4Application
L5DaylightingEPlus	UNIFORMATE10	L5AuxEnergyIncludesDisplayLightGain
L5UseZoneVentilationForMechVent	UNIFORMATE10A	ParamAnalysisType
L5PeakSolarFactor	UNIFORMATE20	ParamAnalysisLevel
ID	UNIFORMATE20A	NCMXMLGenerated
OPTPauseOnError	UNIFORMATF10	L5CalcOptionOverride
OPTServerDownFailQuietly	UNIFORMATF10A	L5OutputOverride
GBXMLImportReportAvailable	UNIFORMATF20	L5HRFreshAirFlowrate
GBXMLImportReport	UNIFORMATF20A	L5HRBypassFreshAirFlowrate
MergeAllowed	UNIFORMATG10	DINBuildingType
IncludeShadingFromExcludedZones	UNIFORMATG10A	DINClimateZone
IncludeShadingFromExcludedZonesHG	UNIFORMATG20	DINNightVent
SSEMultiStateElectrochromicGroupingMethod	UNIFORMATG20A	DINIncludeShading
EDDFile	UNIFORMATG30	DINShading
RDDFile	UNIFORMATG30A	DINDetailedOutput
L5RooflightUValueFactor	UNIFORMATG40	DINTargetZoneSelect
L5LimitGroundFloorUVAl	UNIFORMATG40A	DSMIncludeModelInXML
HVACMethod	UNIFORMATG90	GBXMLAssemblyInstanceExport
NRMCostToxic	UNIFORMATG90A	DSMUpdateEPCCGenImprovements
NRMCostToxicA	UNIFORMATZ10	InterZoneAirflowDefinition
NRMCostMajorDemo	UNIFORMATZ20	HVACHeatingCoilType
NRMCostMajorDemoA	UNIFORMATZ30	HGPeakCoolingLoad
NRMCostSpecGround	NumberCellsBetweenLevels	HGPeakCoolingTime
NRMCostSpecGroundA	CFDMeshingBCFDKNumberOfCores	HGPeakCoolingMonth
NRMCostTempWork	MaximumLocalCells	SBEMS63Stage
NRMCostTempWorkA	MaximumGlobalCells	SBEMS63RRNEPC
NRMCostSiteInvest	MinimumRefinementCells	SBEMS63BSaleDate
NRMCostSiteInvestA	ResolveFeatureAngle	SBEMS63PlanCompDate
NRMCostStairs	NumberSmoothPatchIterations	SBEMS63ActCompDate

SBEMS63PlanCommisBy
SBEMS63ONAME
SBEMS63DelegProt
SBEMS63DelegProtDate
SBEMS63OperatRating
SBEMS63DECLodgDisp
SBEMS63AcceptPres
SBEMS63AltName
SBEMS63AltScenario
SBEMS63AltScenarioDate
SimFourRotatedBaselineBuildings
S63BoilerGt15
S63EditModel
DataPlotOpeningFlowIn
DataPlotOpeningFlowOut
ZoneLabelSize
ZoneLabelSizeMultiplier
IncludeAllBuildingsForEPPolygonLists
CFDSimulationThermalComfortCalculation
CFDComfortActivityMetabolicRate
CFDComfortClothingLevel
CFDComfortRelativeHumidity
CFDSimulationMoistureTransport
CFDSimulationContaminantTransport
CFDSimulationContaminantInitialValue
CFDSimulationContaminantTurbulentSchmidtNumber
CFDSimulationContaminantMolecularDiffusivity
SiteRotation
BuildingRotation
SBEMDistrictHeatingRenewFactor
CFDDisplayMinimumMoistureMassFraction
CFDDisplayMaximumMoistureMassFraction
CFDDisplayMoistureMassFractionContourBandActive1
CFDDisplayMoistureMassFractionContourBandValue1
CFDDisplayMoistureMassFractionContourBandActive2
CFDDisplayMoistureMassFractionContourBandValue2
CFDDisplayMoistureMassFractionContourBandActive3
CFDDisplayMoistureMassFractionContourBandValue3
CFDDisplayMoistureMassFractionContourBandActive4
CFDDisplayMoistureMassFractionContourBandValue4
CFDDisplayMoistureMassFractionContourBandActive5
CFDDisplayMoistureMassFractionContourBandValue5
CFDDisplayMoistureMassFractionContourBandActive6
CFDDisplayMoistureMassFractionContourBandValue6
CFDDisplayMoistureMassFractionContourBandActive7
CFDDisplayMoistureMassFractionContourBandValue7
CFDDisplayMoistureMassFractionContourBandActive8
CFDDisplayMoistureMassFractionContourBandValue8
CFDDisplayMoistureMassFractionContourBandActive9
CFDDisplayMoistureMassFractionContourBandValue9
CFDDisplayMoistureMassFractionContourBandActive10
CFDDisplayMoistureMassFractionContourBandValue10
CFDDisplayMoistureMassFractionContourBandActive11
CFDDisplayMoistureMassFractionContourBandValue11
CFDDisplayMoistureMassFractionContourBandActive12
CFDDisplayMoistureMassFractionContourBandValue12
CFDDisplayMinimumContaminantMassFraction
CFDDisplayMaximumContaminantMassFraction
CFDDisplayContaminantMassFractionContourBandActive1
CFDDisplayContaminantMassFractionContourBandValue1
CFDDisplayContaminantMassFractionContourBandActive2
CFDDisplayContaminantMassFractionContourBandValue2
CFDDisplayContaminantMassFractionContourBandActive3
CFDDisplayContaminantMassFractionContourBandValue3
CFDDisplayContaminantMassFractionContourBandActive4
CFDDisplayContaminantMassFractionContourBandValue4
CFDDisplayContaminantMassFractionContourBandActive5
CFDDisplayContaminantMassFractionContourBandValue5
CFDDisplayContaminantMassFractionContourBandActive6
CFDDisplayContaminantMassFractionContourBandValue6
CFDDisplayContaminantMassFractionContourBandActive7
CFDDisplayContaminantMassFractionContourBandValue7
CFDDisplayContaminantMassFractionContourBandActive8
CFDDisplayContaminantMassFractionContourBandValue8
CFDDisplayContaminantMassFractionContourBandActive9
CFDDisplayContaminantMassFractionContourBandValue9
CFDDisplayContaminantMassFractionContourBandActive10
CFDDisplayContaminantMassFractionContourBandValue10
CFDDisplayContaminantMassFractionContourBandActive11
CFDDisplayContaminantMassFractionContourBandValue11
CFDDisplayContaminantMassFractionContourBandActive12
CFDDisplayContaminantMassFractionContourBandValue12
CFDDisplayVariableMoistureMassFractionContours
CFDDisplayVariableContaminantMassFractionContours
CFDDisplayVariableFilledMoistureMassFractionContours
CFDDisplayVariableFilledContaminantMassFractionContours
CFDDisplay3DContourMoistureMassFraction
CFDDisplay3DContourContaminantMassFraction
ParametricOutput
OPTConvergenceGens
CFDDisplayVariableVelocity
CFDDisplayVariableTemperature
CFDDisplayVariablePressure
CFDDisplayVariableMoisture
CFDDisplayVariableContaminant
CFDDisplayVariableAgeOfAir
CFDDisplayVariablePMV
CFDDisplayVariablePPD
CFDDisplayVariableMeanRadiantTemperature
CFDDisplayVariableOperativeTemperature
NeedUpdateBuildingLevelAreaVolume
HVACZoneHighTemperatureRadiantHeatingRadiantSurface
List