D2.2 – Advanced EU Kernel EPC Engine

OpenBEP4EU is Co-funded by the European Union under Grant Agreement No. 101167613. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA.

DELIVERABLE FACTSHEET

Deliverable no.	**Deliverable D2.2**
Responsible Partner	2 – QUE
WP and Task no. and title	WP2 – Implementation, delivery and demonstration of the EU Kernel for EPC asset calculation T2.2 – Development of the backend of an EU Kernel EPC Engine
Version	1.0
Version Date	15/05/2026

Dissemination level	
X	PU = Public
	PP = Restricted to other programme participants (including the EC)
	RE = Restricted to a group specified by the consortium (including the EC)
	SEN = sensitive
	CO = Confidential, only for members of the consortium (including the EC)

Approvals	**Company**
Authors	Kalamaris Thanos (QUE), Stratis Antonis (QUE), Giannakis Giorgos (QUE), Boubouzioti Anna (QUE), Papanikolaou Antonis (QUE), Andrei Vladimir Litiu (EPB), Dick van Dijk (EPB), Dirk Van Orshoven (EPB), Gerhard Zweifel (EPB), Jaap Hogeling (EPB), Laurent Socal (EPB)
Task Leader	QUE
WP Leader	QUE
1st Reviewer	ED
2nd Reviewer	EUP

ABREVIATIONS

AC	Air Conditioning
API	Application Programming Interface
BEP	Building Energy Performance
BIM	Building Information Model
DHW	Domestic Hot Water
DTV	Design Transfer View
EPB	Energy Performance of Buildings
EPBD	Energy Performance of Buildings Directive
EPC	Energy Performance Certificates
EPW	EnergyPlus Weather
IDS	Information Delivery Specification
IFC	Industry Foundation Classes
JSON	JavaScript Object Notation
REST	Representational State Transfer
TABS	Thermally Activated Building Systems
VRV	Variable Refrigerant Volume
WPs	Work Packages
XML	Extensible Markup Language

PARTNERS SHORT NAMES

Partner No.	Short Name	Full Name
1	**ED**	European Dynamics
2	**QUE**	QUE TECHNOLOGIES
3	**EUP**	EUPHYIA-TECH
4	**SBC**	SBCGR (Sustainable Buildings Council Greece)
5	**EPBC**	EPB CENTER
6	**ENERSAVE**	EnerSave Capital
7	**EU.BAC**	EU Building Automation & Controls Association
8	**EGC**	European Green Cities
9	**ENEFEU**	Energy Efficiency for Europe

About openBEP4EU

*The **openBEP4EU** project is co-funded by the European Union under Grant Agreement No. 101167613 (Life Programme). The project started as of September 1st, 2024 and is anticipated to conclude in March 2027 (30 months duration).*

The building sector is recognised as one of the pillars for achieving the target of European climate neutrality by 2050. The EU is undergoing a twin transition, a digital and green transition as outlined in recent European directives. Buildings are a significant contributor of emissions and they consume a sizeable portion of EU's energy consumption. Towards EU's direction of improving the performance of its building stock, Energy Performance Certificates (EPCs) have, among other initiatives and actions, been instrumental in achieving the set targets. EPCs have become a powerful tool for the end-users, such as service providers who can identify business opportunities, or policymakers who are empowered with better data on the building stock, enabling them to oversee the effects of policies and financial schemes.

A more harmonised European calculation methodology for the EPCs could, among others, increase comparability between regions, trust and market uptake for features related to smart readiness, comfort, real energy consumption and district energy. To this end, OpenBEP4EU aims to deliver an open-source, universal software implementation of the ISO 52000 standards family, namely the EU Kernel EPC Engine, to facilitate simultaneous and coordinated adoption of an innovative EPC calculation approach among all the Member States. This initiative aims to make building performance data easily accessible to various stakeholders like financial institutions, energy service providers and building owners, fostering the development of financial products supporting sustainable renovation projects.

Core elements of openBEP4EU's activities also include the creation of an EPC Support Team to drive market adoption as well as a Sustainable Design Data Hub, aligned with the overarching goals of the European Bauhaus initiative, that not only serves as a data repository but also promotes collaboration and research for sustainable architecture and design. These activities will facilitate stakeholder engagement, contribute to improved EPC reliability, accuracy, accessibility, while paving the way for EPC market uptake across the EU Member States though well promotion strategies. All data sharing within this framework adheres to the principles of the Common European Energy Data Space, prioritizing sovereign data sharing and respecting data ownership among the involved stakeholders.

Executive summary

Deliverable D2.2 vision is to document the development of a harmonised, open-source and standards-compliant calculation kernel capable of implementing the ISO 52000 Standards family in a consistent and reusable manner. The Advanced EU Kernel EPC Engine, hereafter openBEP4EU Calculation Engine, does not seek to replace national EPC schemes or certification procedures. Instead, it provides a reference implementation of the current CEN/ISO EPB calculation standards. Providing a transparent reference implementation along with an evolving set of reproducible test cases and validation examples, the engine aims to enable quality assurance and comparability checks.

The development of the Calculation Engine was based on a systematic assessment of the programmability and relevance of the EPB standards. This assessment evaluated technical maturity, algorithmic determinism, input data availability and impact on EPC indicators. Standards identified as both programmable and relevant to the EPC calculation chain were prioritised for implementation within the engine's scope. The current version implements the core EPB calculation chain, including building thermal envelope modelling (EN ISO 6946, EN ISO 13370, EN ISO 13789), external climatic data processing (EN ISO 52010-1), indoor environmental parameters (EN 16798-1), hourly energy needs calculation (EN ISO 52016-1) and technical system modelling for heating, cooling and domestic hot water (EN 15316 series, ISO 52031, ISO 52032-1). Additional programmable standards have been prioritised for subsequent development phases to progressively expand system coverage.

The findings documented in this deliverable demonstrate that EPB standards can be translated into an executable, modular and scalable software. Preliminary testing and comparative assessment activities, including comparison against EPB reference spreadsheets and EnergyPlus simulation engine, confirm the technical consistency of the implemented modules. The core modules are accessible through the corresponding GitHub repository: Que-Technologies/openbep4eu-calculation-engine.

TABLE OF CONTENTS

ABOUT OPENBEP4EU **4**

EXECUTIVE SUMMARY **5**

1 INTRODUCTION **9**

1.1 STRUCTURE OF DOCUMENT 11
1.2 TARGET AUDIENCE 11
1.3 DISCLAIMER 11

2 FROM EPB STANDARDS TO SOFTWARE IMPLEMENTATION **12**

2.1 CONCEPTUAL FLOW OF ISO 52000 FAMILY EPB STANDARDS 12
2.2 STANDARDS PROGRAMMABILITY ASSESSMENT 14
2.3 STANDARDS ANALYSIS AND ASSESSMENT 16

3 OPENBEP4EU CALCULATION ENGINE **19**

3.1 SYSTEM ARCHITECTURE 19
3.2 DATA IMPORT 21
3.3 DATA VALIDATION 31
3.4 DATA TRANSFORMATION 34
3.5 BUILDING PERFORMANCE SIMULATION: CORE MODULES 36
3.6 BUILDING PERFORMANCE SIMULATION: ORCHESTRATION 39
3.7 TECHNOLOGY STACK 39
3.8 INSTALLATION AND LICENSE 40
3.9 ASSUMPTIONS AND RESTRICTIONS 41

4 TEST CASES AND PRELIMINARY RESULTS **44**

4.1 FREE-FLOATING SINGLE-ZONE THERMAL RESPONSE 46
4.2 SINGLE-ZONE THERMAL ENERGY NEEDS 47
4.3 MULTI-ZONE THERMAL ENERGY NEEDS 49
4.4 SINGLE-ZONE AIR CONDITIONING SYSTEM 53
4.5 MULTI-ZONE AIR-CONDITIONING SYSTEM 55
4.6 MULTI-ZONE HEATING AND DOMESTIC HOT WATER WITH GAS BOILER 56

5 CONCLUSIONS **61**

6 REFERENCES **62**

7 ANNEX A **63**

7.1 ANNEX A.1 – INPUT VARIABLES 63
7.2 ANNEX A.2 – OUTPUT VARIABLES 72

LIST OF FIGURES

Figure 1 – Position of the OpenBEP4EU Calculation Engine within the EPC generation chain, linking structured building and system inputs to calculation-dependent EPC indicators 10
Figure 2 – openBEP4EUCalculation Engine: Standards Flow 13

Figure 3 – Example of the spreadsheet-based traceability structure for the analysis of equations, variables, and final inputs across interrelated EPB standards 17
Figure 4 – Heat-map view of implementation progress across the EPB standards addressed in the current release of the OpenBEP4EU Calculation Engine 18
Figure 5 – OpenBEP4EU Calculation Engine: high-level system architecture 19
Figure 6 – Verification in Solibri Anywhere that the user-defined property sets introduced through the Revit IFC export configuration are present in the IFC4 Add2 DTV file and attached to the intended IFC entities 26
Figure 7 – Rule-based transformation of the validated IFC model into the OpenBEP4EU JSON data model, illustrated for the simplified geometrical input 30
Figure 8 – Geometrical representation of the single-zone configuration used in the preliminary assessment 45
Figure 9 – Geometrical representation of the multi-zone configuration used in the preliminary assessment . 46
Figure 10 – Internal and operative temperatures for the free-floating single-zone test case without window 46
Figure 11 – Internal and operative temperatures for the free-floating single-zone test case with window 47
Figure 12 – Internal and operative temperatures for the single-zone test case (conditioned) 47
Figure 13 – Single-zone test case, Heating and Cooling needs: Comparison with EPB spreadsheets 48
Figure 14 – Single-zone test case, Heating and Cooling needs: Comparison with EnergyPlus 48
Figure 15 – Multizone internal and operative temperatures vs external temperature and GHI (15th of April) . 49
Figure 16 – Multizone internal and operative temperatures vs (ideal) Heating/Cooling load (15th of April) 50
Figure 17 – Multi-zone test case, Heating and Cooling needs: Comparison with EPB spreadsheets 51
Figure 18 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Athens) 51
Figure 19 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Madrid) 52
Figure 20 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Berlin) 53
Figure 21 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Milan) 53
Figure 22 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Stockholm) .. 53
Figure 23 – Single-zone implementation top-down view with air conditioning system 53
Figure 24 – Single-zone test case with air conditioning system: electric energy consumed by the system for heating and cooling 54
Figure 25 – Multizone implementation top-down view with A/C splits 55
Figure 26 – Multi-zone test case with air conditioning system: electric energy consumed by the systems for heating and cooling 55
Figure 27 – Annual share by type of energy needs 56
Figure 28 – Top-down view of the multi-zone heating and domestic hot water with gas boiler test case 56
Figure 29 – Multi-zone heating and domestic hot water with gas boiler test case: emission, distribution, storage, and generation results 58
Figure 30 – Multi-zone heating and domestic hot water with gas boiler test case: Monthly allocation of total energy consumption across thermal zone demands and subsystem contributions 59
Figure 31 – Multi-zone heating and domestic hot water with gas boiler test case: Annual allocation of total energy consumption across thermal zone demands and subsystem contributions 59

LIST OF TABLES

Table 1 – Programmable and non-programmable standards in 2 Clusters 15
Table 2 – Input parameters of the modified BESTEST 600 case 45
Table 3 – Heat pump properties 54
Table 4 – Multi-zone heating and domestic hot water with gas boiler test case: Input parameters 57

LIST OF LISTINGS

Listing 1 – Object-based representation of zone, construction, and surface relationships 22
Listing 2 – Example of required properties in the OpenBEP4EU JSON schema 23
Listing 3 – Example of anyOf allowing alternative parameter representations 23
Listing 4 – Example of oneOf for alternative geometry definitions 23
Listing 5 – Example of Conditional completeness using allOf and anyOf 24
Listing 6 – Example of constrained reference with semantic condition 24
Listing 7 – Example of schedule reference to allowed schedule collections 24
Listing 8 – Fragment of the OpenBEP4EU TXT file for Revit to enable user defined property sets export 25
Listing 9 – Fragment of the OpenBEP4EU IDS for the space properties requirements 27
Listing 10 – Fragment of the OpenBEP4EU IDS for the window material properties requirements 28
Listing 11 – Condensed report in JSON format of the validation of the exported IFC against explicit information requirements specified in the OpenBEP4EU IDS 28
Listing 12 – Example of invalid reference to a non-existing object 32
Listing 13 – Example of a reference which exists but violates semantic constraint 32
Listing 14 – Example of a thermostat definition missing required control information 33
Listing 15 – Example of conflicting geometry representation 34
Listing 16 – Example of invalid schedule reference 34
Listing 17 – OpenBEP4EU Calculation Engine's core modules: thermal_needs.py 37

1 INTRODUCTION

The Energy Performance of Buildings Directive (EPBD) recast [1], Directive (EU) 2024/1275, establishes the regulatory framework for the assessment and certification of building energy performance in the European Union. It defines both the common basis for calculation methodologies and the requirements for EPCs. In particular, Annex I of EPBD requires Member States to describe their national calculation methodologies based on Annex A to the key European standards on the energy performance of buildings, including EN ISO 52000-1, EN ISO 52003-1, EN ISO 52010-1, EN ISO 52016-1, EN ISO 52018-1, EN ISO 52120-1, EN 16798-1, and EN 17423. The same Annex further states that the energy needs and energy use for space heating, space cooling, domestic hot water, ventilation, lighting, and other technical building systems are to be calculated using monthly, hourly, or sub-hourly calculation intervals. The methodology must also represent actual operating conditions and enable the use of metered energy for verification and comparability. The European Commission recommends hourly methods for new buildings, major renovations and buildings with advanced technical systems, renewable energy systems, or energy storage [2].

Automation and control (BACS) is addressed in the EPB framework through EN ISO 52120-1. In practice, the quantitative impact of some control functions is represented within the underlying EPB system modules (e.g., heating, cooling, ventilation, lighting), rather than through a separate “BACS calculation layer”. The current release of the Calculation Engine therefore captures some zone-level control inputs (e.g., heating/cooling setpoints and schedules) and selected system control parameters where these are explicitly supported by the implemented EPB modules. Advanced/supervisory BACS functions (e.g., cross-system coordination and optimisation logic) are not yet represented comprehensively and might not be fully reflected in results. To ensure transparency, a function-coverage register is recommended as a next step (see Assumptions and restrictions / Conclusions).

This regulatory framework is directly linked to the content of EPCs. Under the EPBD recast, EPCs are no longer limited to a single energy class label. They must report a broader set of indicators describing the energy and environmental performance of the building. Article 19 requires the certificate to indicate the energy performance of a building or building unit according to the national methodology adopted under the Directive, while Annex V specifies the information to be displayed [3], including the energy class, primary energy, final energy, the share of on-site renewable energy, operational greenhouse gas emissions, and life-cycle global warming potential where applicable. The energy class remains nationally determined, since Member States define the class thresholds and the related benchmarks within their own implementation frameworks.

Although the Directive provides a common framework, implementation remains nationally differentiated. National methodologies differ in their coverage of EPB standards, in the calculation intervals they use, and in the way they calculate EPC indicators. This diversity was also evident in the technical exchanges carried out by QUE and EPBC with national authorities and representatives in eight Member States during 2024–2025. The same broader picture is reflected in the work of the Concerted Action EPBD [4], [5], which documents the continuing diversity of national approaches to calculations, primary energy indicators, certification practice, and related methodological choices.

This shapes the context in which the OpenBEP4EU Calculation Engine has been developed with the purpose to provide a transparent and reusable implementation of the core EPB calculation procedures needed to generate the principal EPC energy indicators from a common computational basis. As illustrated in Figure 1, the engine is positioned in the certification chain between structured building and system inputs on the one hand, and EPC-relevant outputs on the other. It takes as input building envelope characteristics, thermal zones, occupancy patterns, climatic data, and technical system specifications, and applies an hourly calculation workflow consistent with the EPB family of standards. The resulting outputs include energy needs per thermal zone, energy use per system, final energy use by carrier, primary energy, operational greenhouse gas emissions, and the contribution of on-site renewable energy. These are the calculation-dependent quantities required to

populate a large part of the EPC content defined under the EPBD recast. A central objective of the project is not only to develop this computational core, but also to make it reusable beyond the project. For that reason, the OpenBEP4EU approach foresees the public release, through the project GitHub repository, of a library of core calculation modules implementing the main calculation procedures of the engine. These modules constitute the principal reusable software contribution of the work. They are complemented by repository-level dependency specifications and accompanying documentation so that the released components can be inspected, reproduced, and integrated more easily by external users and developers. This public release focuses on the methodological core of the engine, while surrounding layers related to orchestration, helper-based input preparation, validation services, and broader platform integration remain more closely associated with the project-specific implementation. Within its present scope, the engine is designed to calculate the core energy quantities needed for EPC generation and related assessment workflows. More specifically, it covers the calculation-dependent part of four of the six mandatory front-page EPC indicators introduced in Annex V, namely primary energy, final energy, the share of on-site renewable energy, and operational greenhouse gas emissions, while also producing additional system-level outputs such as energy needs and energy use per technical system. Some EPC elements remain outside the scope of the calculation engine. Energy class assignment depends on national class thresholds established by Member States. Likewise, reference-value comparisons for minimum energy performance standards, nearly zero-energy building levels, or zero-emission building requirements depend on nationally defined benchmarks. Life-cycle global warming potential also follows a distinct methodological track under the Directive and related Commission acts and guidance and is not part of the current calculation-engine scope.

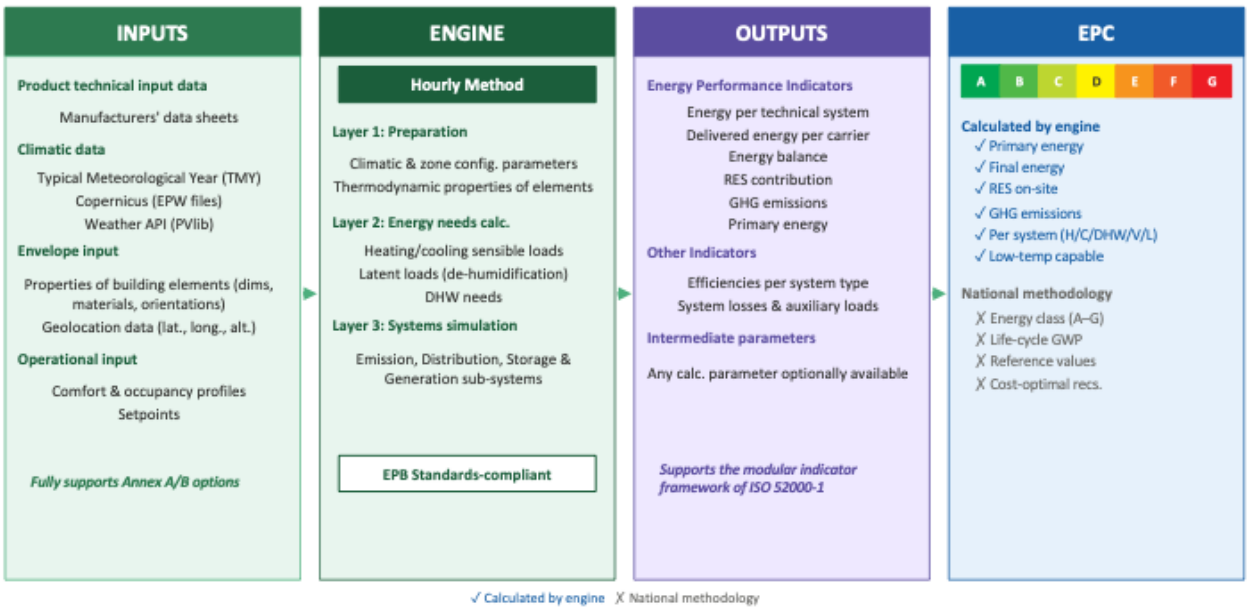


Figure 1 – Position of the OpenBEP4EU Calculation Engine within the EPC generation chain, linking structured building and system inputs to calculation-dependent EPC indicators

The development of the OpenBEP4EU Calculation Engine was carried out through structured collaboration between EPBC experts and QUE developers. The EPBC experts provided interpretation of the standards, methodological guidance, and means (i.e., spreadsheets) for validation of the implemented formulas and outputs. QUE translated the relevant EPB calculation sequences into executable software, designed the supporting data structures, and implemented the computational workflows required for hourly calculation. The current engine already covers key parts of the assessment chain, including building envelope modelling, climate-data processing, indoor environmental parameters, hourly energy-needs calculation, and the modelling of selected heating, cooling, and domestic hot water systems. The following sections present the programmability assessment, the architecture and data structures of the resulting calculation engine.

1.1 Structure of document

This deliverable is structured into three main sections: Section 2 presents the conceptual and methodological basis of the work, describing the flow of the ISO 52000 family of EPB standards, the methodology used for the programmability assessment of EPB standards, and the resulting standards analysis and assessment that informed the implementation priorities; Section 3 describes the OpenBEP4EU Calculation Engine, including its system architecture, data import, validation and transformation workflows, the building performance simulation modules, the orchestration layer, the technology stack, installation and licensing aspects, and the main assumptions and restrictions of the current implementation; and Section 4 presents the test cases and preliminary results, including both validation and comparative assessment activities for single-zone and multi-zone thermal energy needs calculations, as well as selected air-conditioning and domestic hot water simulation cases. Section 5 concludes the deliverable. The document is complemented by Section 6, which provides the references, and by Section 7, which includes annexed material on input and output variables.

1.2 Target Audience

This document is intended for multiple stakeholder groups involved in building energy performance assessment and EPC implementation across the European Union. The primary audiences include: (1) Software developers and technical teams responsible for implementing or integrating EPC calculation engines in Member State certification systems will find detailed information on the engine's architecture, data model, calculation modules and EPB standard implementation approach; (2) EPB standards experts and building physics specialists can use this document to understand how the ISO 52000 family of standards has been interpreted within the openBEP4EU framework, including the methodological decisions, assumptions and prioritisation criteria applied during implementation; and (3) National EPC authorities and policy-makers seeking to understand the technical capabilities, scope and limitations of an EPB calculation engine will benefit from the description of implemented calculation domains, system coverage and the phased development strategy.

1.3 Disclaimer

OpenBEP4EU is Co-funded by the European Union under Grant Agreement No. 101167613. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

2 From EPB standards to software implementation

The EPBD establishes the regulatory framework within which Member States assess and certify the energy performance of buildings. In this context, the Directive refers to the EPB standards as the technical basis for the calculation methodologies used at national level. The ISO 52000 family, together with the associated EN standards, provides the overarching structure for this framework by linking climate data, building energy needs, technical systems, and energy performance indicators into a coherent calculation chain. From a software-development perspective, however, the standards do not constitute a ready-made implementation blueprint. They are distributed across multiple interrelated documents, combine calculation logic with methodological guidance, and often require cross-referencing between standards to identify the full set of equations, variables, and inputs needed for a given result. For this reason, the development of the OpenBEP4EU Calculation Engine required a structured methodological process to move from documentation to executable software. This section presents that process. It is organised in three parts. First, it introduces the conceptual flow of the EPB standards relevant to the OpenBEP4EU Calculation Engine. Second, it explains the programmability assessment carried out to distinguish between standards that could be translated directly into software and standards that play a supporting but non-computational role. Third, it describes the more detailed analysis used to trace formulas, variables, and inputs across the standards and to define the resulting implementation priorities.

2.1 Conceptual flow of ISO 52000 family EPB standards

Figure 2 presents the conceptual standards flow used in OpenBEP4EU to structure the implementation of the calculation engine. The figure is based on a similar flowchart originally developed in the U-CERT project, which has been revised, extended, and reproduced here to reflect the specific scope and implementation of the OpenBEP4EU Calculation Engine. In its current form, it maps the main dependencies between the EPB standards and identifies the sequence of calculations required to derive the energy quantities that subsequently feed EPC indicators. At a high level, the flow can be understood as a progression from input data to energy needs, to technical systems, and finally to aggregated performance indicators.

Input layer: The first layer gathers the standards and source data that provide the physical, climatic, and operational inputs required by the downstream calculations. A first group of standards defines the physical boundary conditions of the building. EN ISO 13370 provides the methodology for heat transfer to the ground and is relevant for ground-contact elements such as slabs on ground and other floor configurations affected by the thermal interaction with the soil. EN ISO 6946 provides the methodology for determining the thermal resistance and thermal transmittance of opaque building elements, thereby supplying the thermal characteristics of the envelope. In cases where more advanced envelope-related effects are considered, related standards on the thermal behaviour of building components and thermal bridges also support the broader interpretation of building fabric performance. A second group provides the climatic and environmental inputs used throughout the assessment. EN ISO 52010-1 supplies the climatic data and solar-related inputs required by the EPB calculation framework, including weather conditions and solar irradiation parameters. These data are essential for the building thermal balance and the downstream treatment of solar gains and climatic exposure. A third group concerns the operational assumptions of the building. EN 16798-1 provides input assumptions related to use patterns, indoor environmental conditions, schedules, and operational profiles. These assumptions affect occupancy-related loads, setpoints, ventilation assumptions, and the timing of building operation.

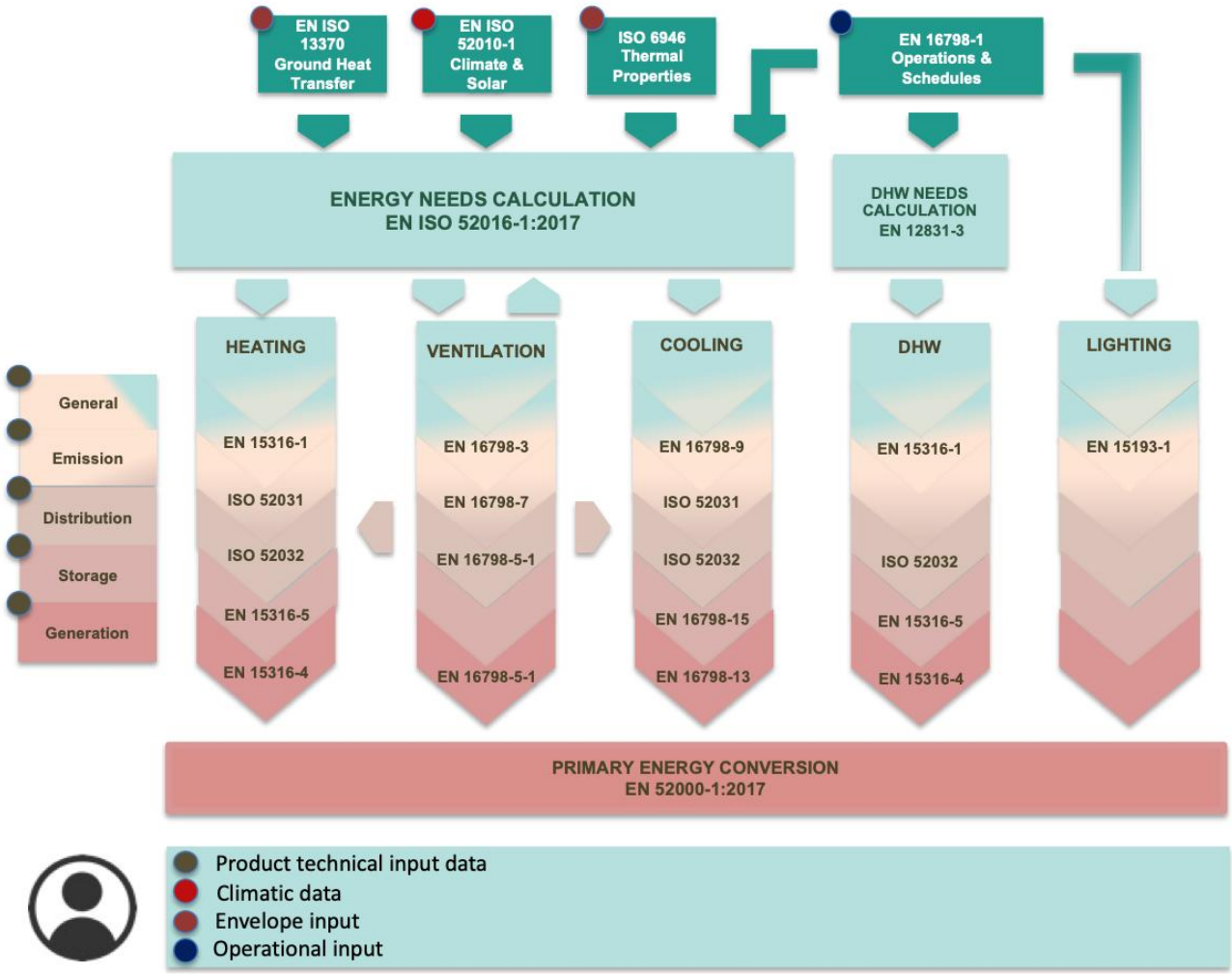


Figure 2 – openBEP4EUCalculation Engine: Standards Flow

Energy needs layer: The second layer concerns the calculation of the building's energy needs, that is, the demand-side quantities that arise from the interaction between the building, its operation, and the external environment. EN ISO 52016-1:2017 provides the core methodology for calculating space heating and space cooling energy needs. Through this standard, the building envelope, ventilation flows, internal gains, climatic boundary conditions, solar gains, and indoor setpoint assumptions are considered to determine the heating and cooling demands of each thermal zone. The same layer is complemented by additional standards addressing other demand categories. EN 12831-3 provides the methodology for calculating the domestic hot water needs. In the conceptual flow shown in Figure 2, ventilation is also represented at the level of needs, since airflow-related behaviour and associated thermal effects influence the overall demand calculation, even where the detailed modelling of ventilation systems is treated later in the systems layer. The output of this second layer is the set of demand-side quantities that the technical building systems are expected to meet, namely the energy needs for heating, cooling, domestic hot water.

Technical systems layer The third layer converts the calculated energy needs into energy use by modelling the operation and performance of the technical building systems. This is the layer in which the EPB framework moves from the demand side of the building to the actual behaviour of the installed systems. In Figure 2, this layer is organised both by system domain and by the standard EPB subsystem structure of general, emission and control, distribution, storage, and generation. For heating systems, the flow includes EN 15316-1 as the general framework and operational related aspects, ISO 52031 emission and control, and ISO 52032-1 for

distribution, EN 15316-5 for storage, and EN 15316-4 for generation. These standards convert heating needs into the corresponding energy use. For ventilation systems, the flow includes EN 16798-3, EN 16798-7, and EN 16798-5-1. These standards address the modelling of ventilation system behaviour, airflow and ventilation-rate aspects, and performance characteristics of mechanical ventilation systems. They translate ventilation-related needs and operation into system-level energy use. For cooling systems, the flow includes EN 16798-9 as the general framework and for operation and control-related aspects, ISO 52031 for emissions and control, ISO 52032-1 for distribution, EN 16798-15 for storage, and EN 16798-13 for cooling generation and cooling-system typologies. These standards govern how cooling needs are met by the corresponding technical systems and how their efficiencies and subsystem losses are reflected in the energy use. For domestic hot water systems, the flow again includes EN 15316-1, ISO 52032-1, EN 15316-5, and EN 15316-4, since DHW is likewise represented through the same broad subsystem logic of general treatment, distribution, storage, and generation. For lighting systems, EN 15193-1 calculates the energy use of lighting systems based on the installed power. This layer is important from a software point of view, since it is the part of the EPB framework where the needs-side results are transformed into actual energy use by carrier. It is also the layer in which the configuration and efficiency of the systems themselves become critical for the final performance indicators.

Output layer The final layer converts the results of the systems calculations into the aggregated performance indicators required by the EPB framework and, ultimately, by EPC generation. At this level, EN ISO 52000-1:2017 plays the overarching role. It provides the framework for combining delivered and exported energy and for converting final energy by carrier into primary energy using national primary energy factors. These factors are defined by Member States for each energy carrier, such as electricity, gas, district heating, or other supplied forms of energy, and may distinguish between renewable and non-renewable contributions. Also other overall indicators can be calculated, such as GHG emissions or energy costs. Hence, the outputs of this layer are the aggregated energy indicators that form the core of the overall energy performance assessment, namely: total primary energy; renewable primary energy; and non-renewable primary energy.

Role of the standards flow in the implementation process The standards flow shown in Figure 2 is not intended only as a conceptual summary of the EPB framework. Within OpenBEP4EU, it also serves as an implementation map. It makes explicit which standards provide inputs, which standards generate intermediate quantities required by others, and which standards are central to the production of EPC-relevant outputs. For that reason, it provided the starting point for the subsequent programmability assessment and for the prioritisation of software development across the standards landscape.

2.2 Programmed and non-programmed standards assessment

Once the conceptual standards flow had been established, the next methodological step was to assess the programmability of the EPB standards. The purpose of this exercise was to determine which standards could be translated directly into software and which, although still important for the overall EPB framework, did not primarily provide executable calculation content. The EPB standards do not all serve the same role within the assessment framework. Some standards contain equations, stepwise procedures, computational rules, and clearly defined input-output relationships that lend themselves to algorithmic implementation. Others provide methodological guidance, reporting requirements, terminology, inspection procedures, or quality-control frameworks that are essential for interpretation and compliance but are not directly programmable. Hence, a structured assessment was needed to distinguish between these two roles and to define a realistic implementation path for the calculation engine. The assessment was carried out in close collaboration with the EPBC experts, including standards authors. Their contribution was particularly important in clarifying issues related to versioning, standard deprecation, overlaps between documents, and the practical feasibility of implementation. On that basis, the initial conceptual flow was refined and the list of standards relevant to the project was

updated. The standards retained for further analysis were then examined jointly by the EPBC experts and the QUE development team.

For each standard, the assessment considered whether: (1) the document contains explicit mathematical equations or deterministic calculation procedures; (2) the required variables, boundary conditions, and outputs are sufficiently defined to support implementation; (3) the logic can be executed algorithmically without requiring case-by-case expert judgement; (4) the standard is directly part of the calculation chain, or instead serves a supporting role through terminology, inspection, documentation, or methodological guidance; (5) only part of the standard is programmable while other parts remain descriptive. As a summary, the examined standards were grouped into two working clusters, presented in Table 1.

Table 1 – Programmed and non-programmed standards in 2 Clusters

Cluster 1: Programmed standards	Cluster 2: Non-programmed standards
▸ EN 12831-1 & -3:2017 ▸ EN 15316 series (-1, -4-1, -4-3, -4-4, -4-8, 4-10) ▸ EN 15459-1:2017 ▸ EN 16798 series (-1:2019; -7, -9, -13, -15, -5-1, -5-2:2017) ▸ EN ISO 6946:2017 ▸ EN ISO 10077-1:2017 ▸ EN ISO 12631:2017 ▸ EN ISO 13370:2017 ▸ EN ISO 13789:2017 ▸ EN ISO 52000-1:2017 ▸ EN ISO 52010-1:2017 ▸ EN ISO 52016 series (-1:2017; -3:2023; -5:20??) ▸ EN ISO 52022 series (-1, -3:2017) ▸ ISO 52032-1:2022 ▸ ISO 52031:2022 ▸ EN ISO 52120-1:2021 * *Not directly encoded in the system	▸ EN 15193-1:2017+A1:2021 ▸ EN 15316 series (-2, -3, -4-2:2017, 4-5:2017) ▸ EN 15378 series (-1, -3:2017) ▸ EN 16798 series (-3, -17:2017) ▸ EN 16946-1:2017 ▸ EN 16947-1:2017 ▸ EN ISO 10077-2:2017 ▸ EN ISO 10211:2017 ▸ EN ISO 14683:2017 ▸ EN ISO 52000-3:2023 ▸ EN ISO 52003-1:2017 ▸ EN ISO 52007-1 ▸ EN ISO 52011-1 ▸ EN ISO 52016-5 ▸ EN ISO 52017-1:2017 ▸ EN ISO 52018-1:2017 ▸ EN ISO 52127-1:2021 ▸ ISO 17772-1:2017 ▸ EN 16798-3:2017 ▸ EN ISO 13786:2017 ▸ EN 17423:2020 ▸ EN 15232-1:2017 ▸ EN 15500-1:2017 ▸ EN 12098 series (-1, -3, -5:2017)

Cluster 1: Programmable standards This cluster includes standards that contain mathematical equations, structured procedures, algorithmic logic, or clearly defined computational rules that can be translated into executable software. This cluster also includes standards that are only partially programmable, meaning that certain sections can be implemented directly, while other parts depend on external inputs, tabulated values, complementary expert material, or assumptions that fall outside a purely algorithmic interpretation.

Cluster 2: Non-programmable standards This cluster includes standards that do not primarily provide executable calculation logic. These standards typically define terminology, reporting structures, inspection and verification procedures, presentation requirements, or broader methodological guidance. Although they are essential for the correct interpretation and application of the EPB framework, they do not correspond directly to software calculation modules. This cluster also includes standards that are programmable but with the

programmable part already covered in other standards (e.g. the content of EN 15316-2 is covered in the more recent ISO 52031; the content of EN 15316-3 is covered in the more recent ISO 52032-1)

The purpose of this classification was not to reduce the importance of the non-programmable standards, but to establish a technically sound boundary between what could be implemented directly in code and what would instead support the implementation through interpretation, validation, documentation, or quality assurance. In other words, the programmability assessment created the bridge between the normative structure of the EPB framework and the practical requirements of software development. This step was also important for implementation planning. Distinguishing standards that contribute directly to the computational chain from those that play a supporting but non-computational role, the assessment allowed the team to prioritise development effort in a transparent way. In particular, it allowed the project to focus on the standards that are central to the production of EPC-relevant outputs, while still acknowledging the broader methodological framework in which these calculations are embedded.

2.3 Standards analysis and assessment

Following the programmability assessment, the next step was a more detailed examination of the standards retained for implementation. The purpose of this stage was to move from the general identification of programmable content to a structured understanding of the actual calculations to be implemented. In practice, this meant analysing the standards at the level of equations, variables, dependencies, and required inputs, so that the content could be translated into a software-oriented representation. This stage also served a second purpose. Beyond supporting implementation, it provided the basis for prioritisation by clarifying which standards were central to the core calculation chain, which standards depended on other parts of the framework, and which standards required supplementary interpretation before they could be translated into code.

2.3.1 Standards decomposition and variable traceability

Once the list of relevant standards had been stabilised, each standard was examined in detail with a view to implementation. For standards that were directly programmable, EPBC experts also provided supplementary material, useful not only for interpretation, but also as a reference point during validation, since it allowed to compare outputs produced by the OpenBEP4EU Calculation Engine with externally prepared reference calculations.

The detailed analysis of the programmable standards focused on three core questions: (1) Which equations and calculation procedures are defined in the standard? (2) Which variables are required by those equations? (3) From where does each variable originate? The third question proved particularly important. In many cases, a variable appearing in one formula is itself the result of another equation, either within the same standard or in a different one. As a result, the standards could not be examined in isolation. The analysis had to reconstruct chains of dependency across multiple documents, progressively tracing the origin of each quantity until the level of final user input, product data, tabulated values, or climate data was reached.

This process generated a hierarchical, tree-like structure of interdependent variables. To manage this complexity in a transparent and reproducible way, the full workflow was documented in a spreadsheet-based traceability structure. Each equation and each variable was recorded along with the source standard, the relevant section of that standard, the notation used in the normative text, the corresponding variable description, and the final input parameters required for implementation. A colour-coding scheme was adopted to make these relations explicit and easier to follow. As illustrated in Figure 3: (1) grey cells indicate the source standard; (2) pink cells contain the description of the variable under examination; (3) light blue cells contain the variable symbol as used in the standard; (4) light green cells indicate the section or equation from which the variable is derived; (5) yellow cells indicate the final required user input; and (6) orange cells indicate that the resulting variable is analysed elsewhere.

This traceability exercise was essential for the software design process. It enabled the development team to move from dispersed normative equations to a structured representation of the data dependencies that the engine needed to support. It also made it possible to identify reusable calculation blocks, clarify which inputs would ultimately need to be provided by the assessor or by product data, and determine where implementation assumptions or expert-supported operationalisations would be required.

Figure 3 – Example of the spreadsheet-based traceability structure for the analysis of equations, variables, and final inputs across interrelated EPB standards

2.3.2 Implementation prioritisation and progress status

The results of the standards analysis were then used to define implementation priorities. The principal driver of that prioritisation was the development of the Building Thermal Simulator, since the calculation of thermal energy needs forms the core of the EPB assessment chain and constitutes a prerequisite for most downstream technical-systems calculations. For that reason, EN ISO 52016-1 was treated as the principal entry point for implementation. Once the thermal-needs engine had been established, the integration of standards related to technical building systems proceeded progressively, focusing first on the domains most relevant to the project scope, namely heating, cooling, and domestic hot water. This sequence reflected both methodological dependency and practical implementation logic: the systems calculations can only be meaningfully developed once the needs-side quantities they are meant to serve are already available. As a result, the current maturity level is not uniform across all standards. It varies depending on the role of each standard in the overall calculation chain, the complexity of its methodology, and the extent to which it has already been integrated and verified within the engine.

For EN ISO 52000-1, the overarching framework has guided the structure of the implementation, but the complete overall assessment chain is not yet fully addressed, since the final primary-energy and aggregated-indicator layer depends on the continued extension of the technical-systems coverage.

For EN ISO 52010-1, the climatic and solar-input aspects required by the current release have been implemented, with the main exception of the shading-related parts, which are still pending.

For EN ISO 52016-1, the hourly calculation methodology has been implemented as the core of the Building Thermal Simulator. The model already covers the essential hourly calculation chain, while some aspects, such as the full validation of latent-load-related results against EPB spreadsheet references, are still being completed.

For EN 16798-1, the elements related to building operation, use assumptions, and schedules have been incorporated, while other parts of the standard, including broader indoor-air-quality and lighting-related aspects, remain outside the currently implemented subset.

For EN ISO 13370, implementation has so far focused on the slab-on-ground case. Other ground-related configurations and related aspects remain outside the scope of the present release.

EN ISO 6946 and EN 12831-3 are considered fully addressed within the current project scope. Likewise, the relevant elements of ISO 52031, ISO 52032-1, EN 15316-5, and EN 15316-4 have been implemented, in several cases using more detailed interpretations agreed with the EPBC experts.

Other standards remain at earlier stages of examination or partial implementation. For instance, within EN 16798-13, only selected cooling-system cases have so far been modelled, while the broader range of cooling-system typologies will be addressed in subsequent releases. Similarly, for EN 16798-7, the natural-ventilation formulations have been reviewed but are not yet fully integrated into the present release, while related mechanical-ventilation calculations under EN 16798-5-1 are still under examination.

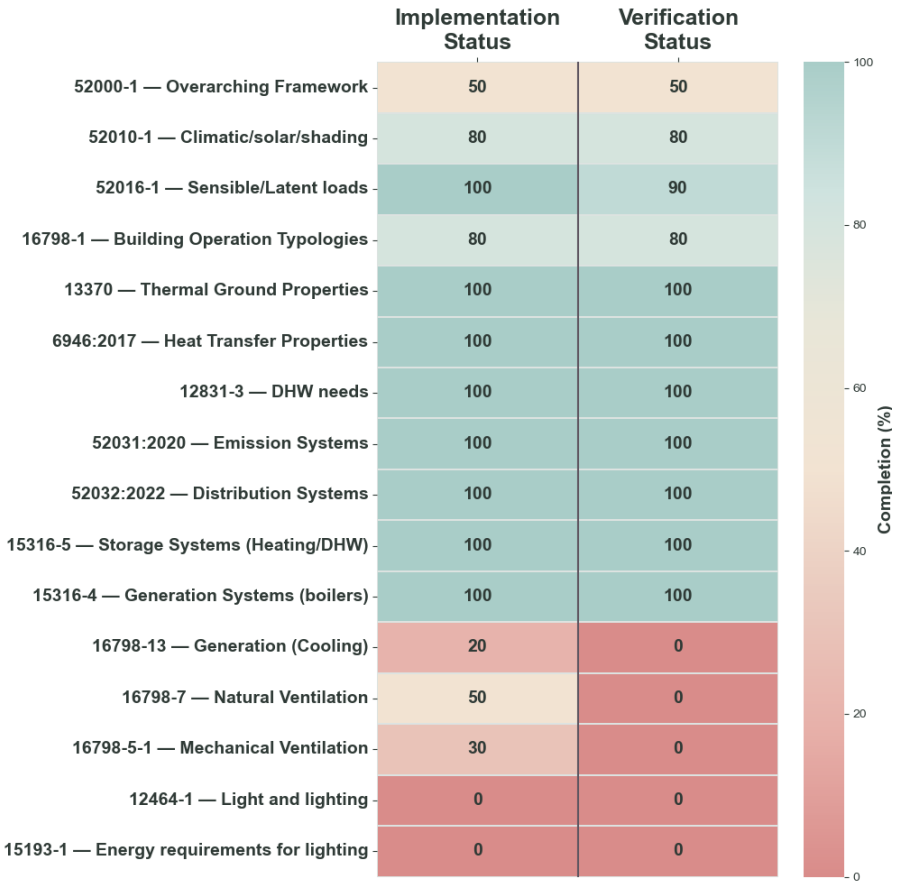


Figure 4 – Heat-map view of implementation progress across the EPB standards addressed in the current release of the OpenBEP4EU Calculation Engine

To provide an overall view of progress, Figure 4 presents a heat-map-style representation of the implementation and verification maturity across the analysed standards. The purpose of the figure is not to imply uniform progress across all parts of the EPB framework, but rather to make the current implementation status transparent and to show how development effort has been prioritised across the standards most relevant to the current release of the OpenBEP4EU Calculation Engine.

3 OpenBEP4EU Calculation Engine

The OpenBEP4EU Calculation Engine has been developed as a modular system, with the main responsibilities of the workflow organised into separate and interoperable components. This choice supports maintainability, makes the implementation easier to extend, and helps keep the overall design clear as new functionalities are added. It also reflects the nature of building energy performance calculation, which is a sequence of interdependent steps, starting from data ingestion and ending with the coordinated execution of domain-specific calculation procedures.

A key advantage of this approach is that it separates the software components that implement the actual calculations from those that deal with workflow management, data preparation, validation, and integration. This distinction is also important from the perspective of dissemination and reuse. In particular, the building performance simulation core modules, namely the modules implementing the calculation procedures and formulas derived from the relevant standards, form the main reusable software assets of the engine and are the part being made publicly available through the dedicated GitHub repository: Que-Technologies/openbep4eu-calculation-engine. The surrounding layers, including orchestration, helper-based input preparation, validation services, and integration, are more closely associated with the implementation of the engine in the specific context of the OpenBEP4EU project and with its interaction with other project components. As such, they are not the primary focus of public release. This does not reduce the transparency of the methodological contribution, since the standard-based calculation procedures are being made publicly available through the released core modules. At the same time, targeted and guided access to selected surrounding layers may be explored upon request from external groups, where this is relevant for technical evaluation, reuse, or integration activities.

3.1 System architecture

As illustrated in Figure 5, the architecture is structured around five main component groups, introduced here and further analysed in the following sections: data import, data validation, data transformation, building performance simulation core, and building performance simulation orchestration. These components define the complete path from input model acquisition to calculation execution and output handling.

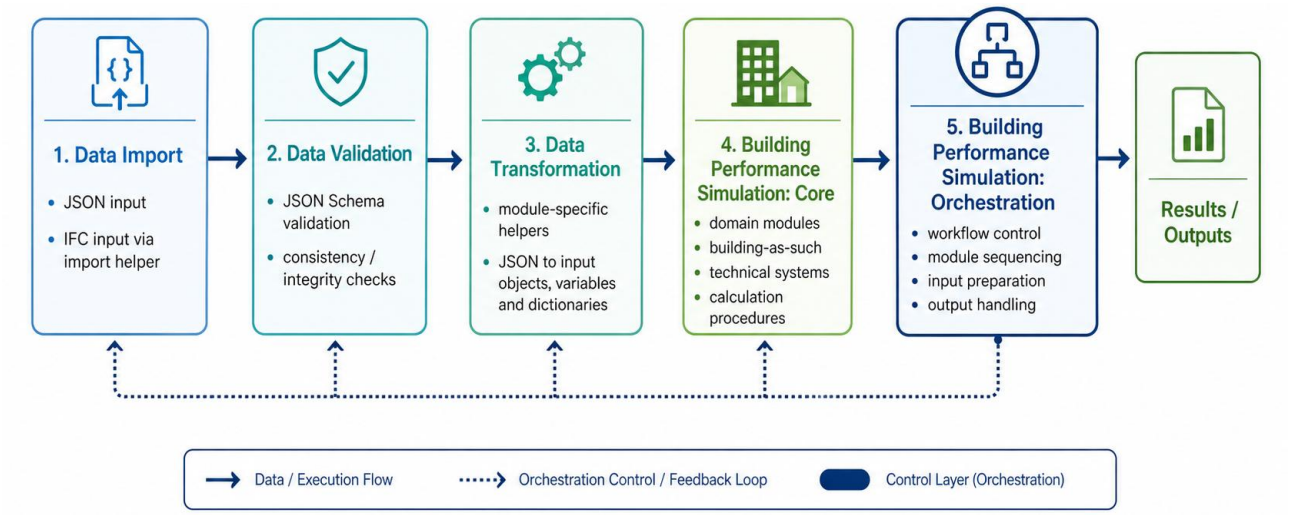


Figure 5 – OpenBEP4EU Calculation Engine: high-level system architecture

Data import The workflow starts at the data import layer. Two main input routes are supported. The first accepts data already provided in the JSON-based structure of the OpenBEP4EU data model (see Section 3.2.1). The

second accepts IFC-based input, in which a dedicated helper extracts the relevant information from the IFC model and converts it into the JSON representation required by the engine (see Section 3.2.2). This dual route is important for interoperability. On the one hand, the engine can operate directly on its own formal internal data model. On the other hand, it can still connect to openBIM-based workflows without requiring upstream systems to produce valid JSON inputs. The IFC route does not introduce an alternative internal representation; instead, it serves as a conversion step that maps external building information into the JSON structure used throughout the engine. This keeps the internal data consistent while allowing the system to work with heterogeneous external sources.

Data validation Once the data have been imported, they are checked by the validation component (see Section 3.3). Its role is to verify that the JSON input complies with the formal rules of the OpenBEP4EU data model and its associated JSON Schema. This includes not only structural checks, such as mandatory properties and value types, but also reference integrity and consistency of the model as a whole. This layer plays a critical role in the overall architecture. It ensures that the rest of the workflow operates on data that are already structurally valid and semantically coherent. Inputs that are incomplete, inconsistent, or malformed are intercepted before they reach the numerical calculation stage. In practice, this makes validation through the quality checker of the engine. It protects the simulation modules from invalid inputs and reduces the risk of unexpected behaviour later in the workflow.

Data transformation After validation, the model passes through a transformation stage in which dedicated helper components derive the specific input parameters needed by the core calculation modules (see Section 3.4). This step is necessary since the OpenBEP4EU data model is designed as a general and object-based representation of the building and its systems, whereas the numerical calculation modules work with more focused and domain-specific inputs. The transformation layer sits between these two levels. It is responsible for extracting the relevant parts of the validated model, resolving the references needed for particular calculations, and packaging the resulting information into the input structures expected by individual modules. Keeping this process in separate helper components is an important part of the modular design, since it avoids coupling the internal data model tightly to the numerical implementation of each module.

Building performance simulation: core modules This layer encapsulates a collection of core modules that implement the actual building performance calculation procedures and formulas derived from the relevant standards and methodologies (see Section 3.5). This is the main technical contribution of the engine. Each module addresses a clearly defined calculation domain or subdomain and works on the prepared inputs produced by the transformation layer. Instead of concentrating the full calculation workflow in a single monolithic solver, the engine breaks it down into separate domain-oriented computational modules. Depending on the workflow, these may cover areas such as envelope-related heat transfer, gains and losses, ventilation-related calculations, control-related behaviour, or other procedures required by the standards being implemented. Structuring the engine in this way makes the software easier to understand, test, validate, and extend. It also means that the main methodological contribution can be shared in a form that is directly reusable by third parties. The simulation core is also where the modularity of the architecture becomes most visible. These modules are specialised computational components with clear responsibilities. This allows to refine a specific calculation domain, add new procedures, or revise one part of the implementation without having to redesign the whole engine.

Building performance simulation: orchestration The orchestration layer manages the execution of the workflow as a whole (see Section 3.6). While the core modules contain the domain-specific calculation methods, orchestration is responsible for deciding how and when those modules are called, how their inputs are prepared, how intermediate results are passed from one step to the next, and how outputs are collected and handled. This separation between calculations and workflow control is essential. The core modules should remain focused on implementing the standardised methods and formulas. The orchestration layer, instead, deals with sequencing, dependencies, exchange of intermediate results, and overall execution control. Handling these

concerns centrally makes the architecture easier to manage and evolve. It also allows to modify the workflow over time, e.g., by adding new modules, changing the order of execution, or adapting the output handling, without having to embed this logic directly inside the simulation core modules.

3.2 Data import

3.2.1 OpenBEP4EU data model

The OpenBEP4EU Calculation Engine is built around a data model that provides a formal, machine-readable, and interoperable representation of building energy performance data, suitable for data entry, automated validation, transformation workflows, and simulation processes. This need arises from a known limitation in the current landscape. On the one hand, regulatory and EPB-oriented methods usually define structured inputs and outputs, yet they often rely on aggregated quantities and implicit assumptions. On the other hand, detailed simulation environments offer far richer modelling possibilities, but they commonly depend on software-specific formats that are difficult to exchange, validate, or embed in wider digital workflows. What is often missing is a formal representation that is structured enough for automation, while remaining sufficiently flexible to support different modelling contexts.

To address this, as presented earlier, a detailed analysis of standards and methodologies was carried out, with particular attention to their interrelations and to the input-output tables through which information moves from one calculation step to the next. The identified and programmed parameters are described in Section 7. The purpose was not simply to collect parameters, but to identify the entities that must exist, the relationships between them, and the way they support the calculation logic across domains such as the building envelope, internal gains, controls, ventilation, and technical systems. The classes included in the data model are the outcome of that work. They are grounded in a structured interpretation of the relevant standards, rather than in a software-specific view of the problem. From a state-of-the-art perspective, the main contribution of the OpenBEP4EU data modelling approach lies in moving from loosely structured input-output descriptions to a formal, schema-driven representation. Conventional formats typically specify which data should be available; however, they do not usually provide a mechanism for checking that data rigorously across structural and semantic dimensions. The OpenBEP4EU Calculation Engine addresses this by defining the data and the rules that govern how valid data must be organised. In that sense, the model does not merely document expectations; it makes them executable. The model also follows an object-oriented logic. A building is not represented as a flat list of parameters, but as a set of explicit and interrelated objects. A zone, a construction, a surface, a schedule, or a thermostat is treated as a separate object with its own properties and explicit links to other objects.

JSON schema: rationale and advantages The OpenBEP4EU data model is defined in JSON format and governed by a formal JSON Schema specification. JSON is lightweight, widely supported, hierarchical, and well suited to represent complex object structures such as buildings and their components. It is readable enough to support manual inspection and debugging, while remaining structured enough for reliable machine processing. It also fits well within modern software environments, including Python, JavaScript, web interfaces, REST APIs, and cloud-based workflows. However, JSON alone would not have been sufficient. The methodological strength of the implemented approach lies in the use of JSON Schema. The OpenBEP4EU JSON Schema has been developed to formally describe the expected structure of the data, the acceptable property types, the mandatory attributes, the logical alternatives, and the relationships between objects. This offers several practical advantages. It makes the data model explicit, enables automatic validation before calculations are executed, supports the generation of guided user interfaces, simplifies transformation pipelines from external sources such as openBIM models, and provides a single reference specification that can be used consistently across software components. It is worth noting that the schema is not an accessory around the calculation engine. It is part of its architecture. It captures both structure and modelling definitions. This includes,

for example, the fact that some properties are always required, some parameters may be expressed in more than one valid form, certain representations are mutually exclusive, and many objects must refer to other objects under specific semantic conditions.

A simple instance of the data model is shown in Listing 1. This is not a schema fragment, but a schema-compliant JSON instance. Its purpose is to make the object-based structure of the model visible in a compact example. The surface `wall_1` does not include all information required for its interpretation in isolation. Instead, it refers to a zone through `zoneId` and to a construction through `constructionId`. This small example illustrates one of the central design choices of the model: information is distributed across linked objects rather than repeated locally. That choice avoids duplication, supports consistency, and allows a single object definition, such as a construction, to be reused wherever needed without creating conflicting copies.

```
{
  "zone": {
    "zone_1": {
      "id": "zone_1",
      "type": "thermal zone",
      "volume": 100.0,
      "floorArea": 40.0
    }},
  "construction": {
    "wall_constr": {
      "id": "wall_constr",
      "constructionClass": "mass equally distributed"
    }},
  "surface": {
    "wall_1": {
      "id": "wall_1",
      "zoneId": "zone_1",
      "constructionId": "wall_constr",
      "outsideBoundaryCondition": "External",
      "area": 10.0,
      "height": 2.5,
      "azimuth_angle": 0,
      "tilt_angle": 90
    }}}
```

Listing 1 – Object-based representation of zone, construction, and surface relationships

The OpenBEP4EU JSON schema is not publicly distributed. The schema is the authoritative internal specification of the calculation engine, and it includes modelling and validation rules that belong to the internal implementation. What will be made public are instantiated input and output models that comply with the schema. These are sufficient to document the practical use of the model, support integration work, and demonstrate the semantics of the classes and relationships. In other words, compliant JSON instances will be publicly available, while the full rule set that governs them remains internal.

JSON Schema: representative fragments/excerpts The full JSON Schema for the OpenBEP4EU Calculation Engine is extensive. It includes the complete set of classes, definitions, property constraints, logical constructs, and cross-object rules needed to support the model. Reproducing the full schema in the present report would not be useful. It would introduce a large amount of technical detail at the expense of readability, and it would obscure the conceptual points that are most important for understanding the design decisions and the added value of the work. For this reason, the following listings are presented as selected excerpts. These excerpts have been chosen to illustrate distinct mechanisms within the schema. They do not define the whole model in isolation; rather, they show how specific modelling requirements are formalised. The selected fragments cover minimum completeness conditions, controlled alternative representations, mutually exclusive modelling choices, compound logical requirements, semantically constrained references, and explicit schedule linkage.

A first basic mechanism is shown in Listing 2, which uses the `required` keyword to specify a minimum completeness condition. The four properties listed here (`id`, `latitude`, `longitude`, and `externalAirAndSkyTemperatureDifference`) define a minimum set of information needed to identify the object and locate it within a climatic context relevant to energy-related calculations. What the fragment makes explicit is that these properties are not optional documentation hints; they are mandatory elements of a valid object. If any one of them is absent, validation fails. This is precisely where the value of the schema becomes evident: requirements that would otherwise remain descriptive are turned into enforceable rules.

```
{"required": ["id", "latitude", "longitude",
              "externalAirAndSkyTemperatureDifference"]}
```

Listing 2 – Example of required properties in the OpenBEP4EU JSON schema

A different mechanism is illustrated in Listing 3, which uses `anyOf` to allow more than one valid representation of the same parameter. The specific excerpt allows the parameter to be expressed either through a controlled textual category (here, a climatic classification such as “sub-polar areas”, “tropics”, or “intermediate zones”) or through a numerical value. This is a good example of controlled flexibility. The schema does not leave the field open to arbitrary content; instead, it recognises that in some modelling situations a standard category is appropriate, while in others a project-specific numerical input is more suitable. The rule therefore preserves flexibility without sacrificing clarity or control over admissible values.

```
{"anyOf": [
           {"type": "string", "enum": ["sub-polar areas", "tropics",
              "intermediate zones"]},
           {"type": "number"}]}
```

Listing 3 – Example of anyOf allowing alternative parameter representations

Listing 4 introduces a stricter logical condition through `oneOf`. Here the issue is not that several representations are acceptable in parallel, but that exactly one modelling approach must be chosen. In this case, the fragment governs how surface geometry is described. A surface may be defined through a simplified set of geometric descriptors (`area`, `height`, `azimuth_angle`, and `tilt_angle`) or through a detailed vertex-based representation. The fragment makes clear that these are alternative modelling routes, not complementary ones. The surface cannot use both at the same time, and it cannot omit both. This is especially important for downstream interpretation, as it avoids ambiguity about which geometric description should be used by calculation or transformation routines.

```
{"oneOf": [
           {"required": ["area", "height", "azimuth_angle", "tilt_angle"]},
           {"required": ["vertices"]}]}
```

Listing 4 – Example of oneOf for alternative geometry definitions

The fragment in Listing 5 shows how the schema handles a more composite form of modelling. The example concerns thermostat definition and combines `allOf` and `anyOf` to express two requirements simultaneously. First, the object must define both heating control and cooling control. Second, each of these may be represented in one of two valid ways: either through a schedule reference or through a constant setpoint. The structure of the fragment mirrors this logic exactly. The outer `allOf` states that both control dimensions are required, while the nested `anyOf` clauses state that either representation is acceptable for each one. This is a more precise rule than a simple list of required fields, as it captures a modelling decision that would otherwise remain embedded in application code or explanatory text.

```
{"allOf": [{
      "anyOf": [
        {"required": ["heatingSetpointSchedule"]},
        {"required": ["constantHeatingSetpoint"]}
      ]},{
```

```
      "anyOf": [
        {"required": ["coolingSetpointSchedule"]},
        {"required": ["constantCoolingSetpoint"]}]}]}
```

Listing 5 – Example of Conditional completeness using allOf and anyOf

An equally important aspect of the model is the way references between objects are constrained. Listing 6 shows that a property such as `zoneId` is not treated as a free identifier. It must conform to the common identifier definition, it must point to an object in the `zone` collection, and it must satisfy an additional semantic condition: the referenced object must have `type = "thermal zone"`. This last condition is particularly important. It means that the schema does not merely check whether a referenced object exists; it also checks whether that object is of the right kind for the relationship being specified. In other words, it prevents a surface involved in thermal calculations from being linked to a zone object that serves another modelling purpose, such as a lighting service area.

```
{"zoneId": {
    "$ref": "#/$defs/id",
    "x-ref-collection": "zone",
    "x-ref-constraint": {
      "property": "type",
      "equals": "thermal zone"}}}
```

Listing 6 – Example of constrained reference with semantic condition

The final schema excerpt, shown in Listing 7, concerns schedule linkage. Here the schema specifies that `heatingSetpointSchedule` must refer to an object belonging to either the `scheduleConstant` or the `scheduleCompact` collection. This is more significant than it may first appear. Time-dependent input is often a source of inconsistency in digital models, particularly when schedules are handled through loosely defined labels or informal conventions. The rule shown here ensures that schedule references always resolve to recognised schedule objects with an explicit place in the model structure.

```
{"heatingSetpointSchedule": {
    "$ref": "#/$defs/id",
    "x-ref-collection": ["scheduleConstant", "scheduleCompact"]}
```

Listing 7 – Example of schedule reference to allowed schedule collections

3.2.2 IFC-based data import

In addition to accepting input directly in the JSON structure of the OpenBEP4EU data model, the calculation engine supports an IFC-based import route. This route has been introduced to connect the engine with openBIM workflows and, in particular, with BIM models authored in common design environments and exported to IFC. In the present implementation, the helper processes any IFC4 Add2 Design Transfer View (DTV) file exported from Revit and converts its content into the corresponding JSON representation required by the calculation engine. For the helper to generate a reliable and semantically consistent JSON model, the IFC input must satisfy a number of preconditions related to modelling practice, property naming, and content completeness. For this reason, the IFC-based import process has been structured into four steps, each addressing a specific requirement of the workflow.

Step 1: Preparation of the BIM model according to modelling guidelines The first step is to ensure that the BIM model is developed in a way that supports downstream processing for energy performance calculation. This is a necessary precondition, since the quality and consistency of the exported IFC depend directly on the way the source BIM model has been authored. In the OpenBEP4EU workflow, this is addressed through modelling guidelines developed in the context of related Horizon Europe projects. These guidelines define the expected modelling conventions, the relevant building elements and relationships, and the information that needs to be present in the authoring model so that it can later be interpreted correctly by the import helper. The purpose of this step is to reduce ambiguity at the source. If modelling choices are inconsistent, incomplete, or not aligned

with the expected information structure, the IFC export may still be formally valid, yet unsuitable for automated mapping into the JSON data model. The guidelines are used to establish a sufficiently controlled BIM authoring process before export takes place.

Step 2: Alignment of property names through a user-defined property set file To ensure that the IFC export contains the properties required by the OpenBEP4EU IFC-based import helper, a user-defined property set text file was prepared and imported into Revit before export. This step is important, since the helper does not rely on arbitrary property names or on project-specific interpretation at import stage. Instead, it expects a controlled set of property-set names and property names that can be mapped directly to the corresponding attributes of the OpenBEP4EU JSON data model. Revit supports this workflow through user-defined IFC property set configuration files, which allow the exporter to assign named properties to specific IFC entity types during export.

A fragment of the property set definition used in the workflow is shown below, where four custom property sets are defined: `ThermalSpace_EnergyAnalysisProperties` for `IfcSpace`, `TransparentElement_EnergyAnalysisProperties` for `IfcDoorType` and `IfcWindowType`, `OpaqueElement_EnergyAnalysisProperties` for `IfcWallType`, `IfcSlabType`, and `IfcRoofType`, and `ThermalZone_EnergyAnalysisProperties` for `IfcZone`. Each property set contains a controlled list of properties relevant to energy-related processing. For example, the space-level property set includes geometric and use-related attributes such as `Height`, `Perimeter`, `Area`, `AreaPerPerson`, `InfiltrationAirChangePerHour`, and `ConditionType`, while the transparent- and opaque-element property sets include thermal attributes such as `ThermalTransmittance`, `SolarHeatGainCoefficient`, `VisibleTransmittance`, `ThermalResistance`, `ThermalMass`, and `Roughness`. The purpose of this configuration is to ensure that the exported property names are already aligned with the semantics expected by the downstream helper.

```
# User Defined PropertySet Definition File
# OpenBEP4EU

PropertySet:      ThermalSpace_EnergyAnalysisProperties     I     IfcSpace
      Height                              Length                  Unbounded Height
      Perimeter                           Length                  Perimeter
      Area                                Area                    Area
      AreaPerPerson                       Area                    Area per Person
      SensibleHeatGainPerPerson           Power                   Sensible Heat Gain per person
      RadiativeEquipmentHeatGainPerArea   Real                    Specified Lighting Load per area
      LatentHeatGainPerPerson             Power                   Latent Heat Gain per person
      ConvectiveEquipmentHeatGainPerArea  Real                    Specified Power Load per area
      InfiltrationAirChangePerHour        Real                    Air Changes per Hour
      Occupied                            Boolean                 Occupiable
      ConditionType                       Text                    Condition Type

PropertySet: TransparentElement_EnergyAnalysisProperties   I     IfcDoorType, IfcWindowType
      ThermalTransmittance            ThermalTransmittance       Heat Transfer Coefficient(U)
      SolarHeatGainCoefficient        Real                       Solar Heat Gain Coefficient
      VisibleTransmittance            Real                       Visual Light Transmittance

PropertySet: OpaqueElement_EnergyAnalysisProperties  I     IfcWallType, IfcSlabType, IfcRoofType
      ThermalTransmittance            ThermalTransmittance       Heat Transfer Coefficient(U)
      ThermalResistance               Real                       Thermal Resistance (R)
      ThermalMass                     Real                       Thermal Mass
      SolarAbsorptance                Real                       Absorptance
      Roughness                       Integer                    Roughness

PropertySet:      ThermalZone_EnergyAnalysisProperties      I     IfcZone
      HeatingTemperatureSetpoint      ThermodynamicTemperature   Heating Set Point
      CoolingTemperatureSetpoint      ThermodynamicTemperature   Cooling Set Point
      HumidificationRelSetpoint       Real                       Humidification Set Point
      DehumidiciationRelSetpoint      Real                       Dehumidification Set Point
```

Listing 8 – Fragment of the OpenBEP4EU TXT file for Revit to enable user defined property sets export

For demonstration purposes, a simple BIM model was authored in Revit following the modelling guidelines described in Step 1. The user-defined property set text file presented above was then imported into the Revit IFC export configuration so that the relevant energy-related parameters would be exported under the expected IFC property set and property names. The model was subsequently exported as an IFC4 Add2 DTV file and inspected in Solibri Anywhere. The screenshots from Solibri Anywhere, illustrated in Figure 6, show that this export step worked as intended. In the exported IFC model, the custom property sets are visible on the expected IFC objects: `ThermalSpace_EnergyAnalysisProperties` appears on space-related objects, `OpaqueElement_EnergyAnalysisProperties` is attached to wall-related objects, and `TransparentElement_EnergyAnalysisProperties` is visible on window-related objects. The screenshots also confirm that the corresponding thermal and use-related values were transferred successfully. These include, for example, area- and occupancy-related properties on spaces, thermal transmittance and resistance on opaque elements, and U-value, solar heat gain coefficient, and visible transmittance on transparent elements.

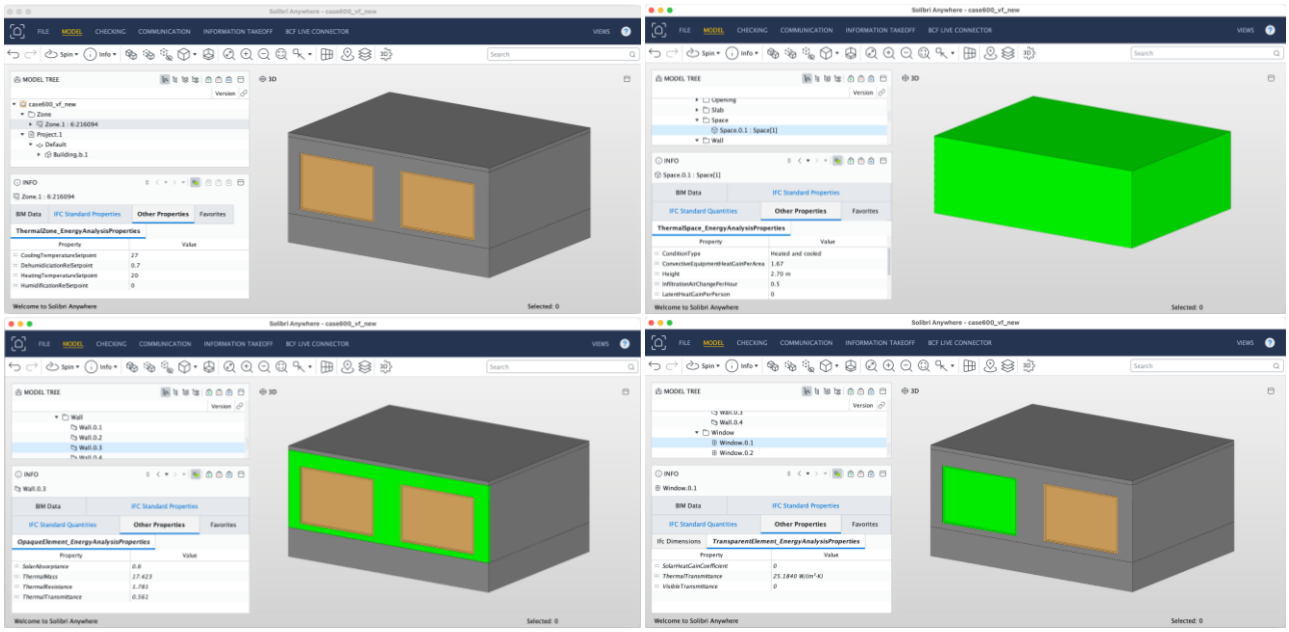


Figure 6 – Verification in Solibri Anywhere that the user-defined property sets introduced through the Revit IFC export configuration are present in the IFC4 Add2 DTV file and attached to the intended IFC entities

Step 3: Validation of the exported IFC against explicit information requirements The third step aims to check whether the exported IFC file contains the information required by the OpenBEP4EU IFC-based import workflow, and whether that information is organised in the expected way. This is an important distinction. A file may be syntactically valid as IFC and still be unsuitable for downstream automated processing if key entities, property sets, or properties are missing, incomplete, or assigned to the wrong IFC classes. To address this, the workflow combines two elements: (1) an Information Delivery Specification (IDS) in XML format, which formally states the information requirements that the IFC file must satisfy; and (2) an IFC validation method, which compares the exported IFC model against that specification.

IDS is a buildingSMART standard for expressing exchange requirements in a computer-interpretable form. It is intended precisely for this use: defining which objects, properties, values, and related conditions must be present in an IFC model so that compliance can be checked automatically. Within the present workflow, the IDS XML acts as the formal description of what the OpenBEP4EU helper expects to receive from the IFC file before mapping begins. The validator then checks whether the exported IFC complies with those requirements. In this way, validation is not limited to generic IFC correctness; it tests whether the file is ready for the specific conversion process of the calculation engine. This makes the step particularly valuable in automated workflows, where missing or misnamed information should be detected before the IFC-to-JSON mapping is attempted.

For the demonstrator introduced in Step 2, a fragment of the complete IDS was aligned with the same four custom property sets exported from Revit through the TXT-based configuration, namely `ThermalSpace_EnergyAnalysisProperties`, `TransparentElement_EnergyAnalysisProperties`, `OpaqueElement_EnergyAnalysisProperties`, and `ThermalZone_EnergyAnalysisProperties`. The fragment reflects all these requirements. However, to keep the presentation concise, only two representative excerpts are shown below.

The first excerpt, shown in Listing 9, concerns `IfcSpace` and checks for the presence of the property set `ThermalSpace_EnergyAnalysisProperties` together with selected properties required by the downstream workflow.

```
<ids:specification name="OpenBEP4EU space requirements">
  <ids:applicability>
    <ids:entity>
      <ids:name>
        <ids:simpleValue>IFCSPACE</ids:simpleValue>
      </ids:name>
    </ids:entity>
  </ids:applicability>
  <ids:requirements>
    <ids:property>
      <ids:propertySet>
        <ids:simpleValue>ThermalSpace_EnergyAnalysisProperties</ids:simpleValue>
      </ids:propertySet>
      <ids:baseName>
        <ids:simpleValue>Height</ids:simpleValue>
      </ids:baseName>
    </ids:property>
    <ids:property>
      <ids:propertySet>
        <ids:simpleValue>ThermalSpace_EnergyAnalysisProperties</ids:simpleValue>
      </ids:propertySet>
      <ids:baseName>
        <ids:simpleValue>Area</ids:simpleValue>
      </ids:baseName>
    </ids:property>
    <ids:property>
      <ids:propertySet>
        <ids:simpleValue>ThermalSpace_EnergyAnalysisProperties</ids:simpleValue>
      </ids:propertySet>
      <ids:baseName>
        <ids:simpleValue>ConditionType</ids:simpleValue>
      </ids:baseName>
    </ids:property>
  </ids:requirements>
</ids:specification>
```

Listing 9 – Fragment of the OpenBEP4EU IDS for the space properties requirements

The second excerpt, shown in Listing 9 concerns `IfcWindowType` and checks for the transparent-element property set and its key thermal-optical parameters.

```
<!-- Simplified illustrative IDS excerpt for IfcWindowType -->
<ids:specification name="OpenBEP4EU transparent element requirements">
  <ids:applicability>
    <ids:entity>
      <ids:name>
        <ids:simpleValue>IFCWINDOWTYPE</ids:simpleValue>
      </ids:name>
    </ids:entity>
  </ids:applicability>
  <ids:requirements>
    <ids:property>
```

```
      <ids:propertySet>
        <ids:simpleValue>TransparentElement_EnergyAnalysisProperties</ids:simpleValue>
      </ids:propertySet>
      <ids:baseName>
        <ids:simpleValue>ThermalTransmittance</ids:simpleValue>
      </ids:baseName>
    </ids:property>
    <ids:property>
      <ids:propertySet>
        <ids:simpleValue>TransparentElement_EnergyAnalysisProperties</ids:simpleValue>
      </ids:propertySet>
      <ids:baseName>
        <ids:simpleValue>SolarHeatGainCoefficient</ids:simpleValue>
      </ids:baseName>
    </ids:property>
    <ids:property>
      <ids:propertySet>
        <ids:simpleValue>TransparentElement_EnergyAnalysisProperties</ids:simpleValue>
      </ids:propertySet>
      <ids:baseName>
        <ids:simpleValue>VisibleTransmittance</ids:simpleValue>
      </ids:baseName>
    </ids:property>
  </ids:requirements>
</ids:specification>
```

Listing 10 – Fragment of the OpenBEP4EU IDS for the window material properties requirements

The validation method developed for this step is based on IfcTester, which supports the use of IDS specifications for IFC checking and the generation of validation reports.

For demonstration purposes, the IFC4 Add2 DTV file exported in Step 2 from the simple Revit model was then checked against the complete IDS specification.


```
{
  "title": "OpenBEP4EU IFC requirements",
  "date": "2026-03-22 12:10:00",
  "filepath": "openbep4eu_requirements.ids",
  "filename": "openbep4eu_requirements.ids",
  "status": true,
  "total_specifications": 4,
  "total_specifications_pass": 4,
  "total_specifications_fail": 0,
  "percent_specifications_pass": 100,
  "total_requirements": 23,
  "total_requirements_pass": 23,
  "total_requirements_fail": 0,
  "percent_requirements_pass": 100,
  "total_checks": 51,
  "total_checks_pass": 51,
  "total_checks_fail": 0,
  "percent_checks_pass": 100
}
```


Listing 11 – Condensed report in JSON format of the validation of the exported IFC against explicit information requirements specified in the OpenBEP4EU IDS

The condensed output shown in Listing 10 summarises a successful IDS-based validation of the demonstrator IFC model. The field `total_specifications` refers to the four specification groups considered in the validation, corresponding to the four custom property sets introduced in the workflow: `ThermalSpace_EnergyAnalysisProperties`, `TransparentElement_EnergyAnalysisProperties`, `OpaqueElement_EnergyAnalysisProperties`, and `ThermalZone_EnergyAnalysisProperties`. The field total_requirements gives the total number of individual property requirements checked across these four groups. In this case, the IDS contains 23 such requirements in total: 11 for the space-related property set, 3 for

transparent elements, 5 for opaque elements, and 4 for thermal zones. The field `total_checks` reflects the fact that each requirement is evaluated against all applicable objects in the IFC model. For the demonstrator, this means one `IfcSpace`, two window objects, six opaque building elements, and one `IfcZone`. As a result, the total number of effective checks is higher than the number of requirements alone. More specifically, the 11 space-related requirements are checked once, the 3 transparent-element requirements are checked for each of the 2 windows, the 5 opaque-element requirements are checked for each of the 6 opaque elements, and the 4 thermal-zone requirements are checked once. This leads to a total of 51 checks. Since all specifications, requirements, and checks passed successfully, the summary reports status = true, with zero failed specifications, requirements, or checks. In practical terms, this means that the demonstrator IFC file satisfied all the information requirements expressed in the IDS and could proceed to the next stage of the workflow, namely mapping by the IFC-based import helper.

Step 4: Mapping of the validated IFC to the OpenBEP4EU JSON data model Once the exported IFC file has passed IDS-based validation, it can be processed by the IFC-based import helper. This fourth step is the actual transformation stage of the workflow. Its purpose is to convert the validated IFC content into the object structure required by the OpenBEP4EU calculation engine. The helper encapsulates a collection of mapping rules that interpret IFC entities, property sets, material definitions, and relationships and translate them into the corresponding classes, attributes, and references of the OpenBEP4EU JSON data model. The Python package utilised for this step is IfcOpenShell, which supports serialisation and deserialisation of IFC entities, their types, properties, relationships, and geometry-related information.

Each rule specifies which IFC pattern is being recognised, which JSON object or objects should be created from it, which properties should populate the target attributes, and how the resulting objects should be linked to one another. The rule groups discussed below is representative subsets of the overall mapping, focusing on the building-geometry, boundary, and construction side of the mapping process, without implying that these are the only rules implemented in the helper. The complete collection is broader and includes dedicated mapping rules for technical systems**,** so that system-related IFC content can likewise be transformed into the corresponding JSON objects required by the OpenBEP4EU data model.

> **spaces and thermal zones**: each `IfcSpace` that enters the import route is assumed to be mapped to a corresponding thermal-zone-level object in the OpenBEP4EU JSON data model. This design assumption is what allows the rest of the workflow to link surfaces, schedules, and control information consistently to a thermal calculation context. Where an `IfcZone` is available and carries the custom property set `ThermalZone_EnergyAnalysisProperties`, it provides the zone-level control attributes, such as heating and cooling setpoints, that are attached to the corresponding JSON zone object. In other words, the mapping does not treat the space merely as a geometric subdivision, but as the basis for a thermal zone representation aligned with the internal logic of the calculation engine.

> **surfaces, their geometry, and their relation to zones**: Here the helper relies primarily on `IfcRelSpaceBoundary2ndLevel` entities and their relationships rather than on isolated element geometry alone. In the OpenBEP4EU import workflow, these relations are used to construct JSON surface and subsurface objects, derive their geometry, establish their link to the corresponding thermal zone, and determine their boundary context. This is a key point for the target data model. In the OpenBEP4EU JSON structure, a surface is not only a geometric entity; it is also an analytical boundary object linked explicitly to a zone and characterised by a specific boundary condition. The mapping logic addresses this by treating the space-boundary relation as the primary source for surface interpretation. When a boundary relates a space to a physical building element, the corresponding JSON surface is created as a physical boundary associated with that zone. When the relation expresses a virtual boundary or an adjacent-space condition, the helper uses that information to assign the appropriate meaning in the target JSON representation.

> **constructions and materials**: Here the helper uses IFC material associations to build the construction-related and material-related objects required by the OpenBEP4EU data model. For layered opaque elements such as walls, slabs, and roofs, material information may be obtained from `IfcMaterialLayerSet`-based definitions, while for assemblies or multi-part elements the mapping may also make use of `IfcMaterialConstituentSet`. The purpose is to translate the structure of the IFC model into JSON construction objects and associated material objects, so that the resulting model preserves both the relationships and the attributes needed by the calculation engine. In practice, this material-association is combined with the custom energy-analysis property sets introduced in Step 2. The IFC material definitions provide the composition structure, while the exported thermal properties provide the calculation-oriented attributes expected by the JSON classes.

> **references between objects**: This is essential for the OpenBEP4EU JSON data model, which is explicitly object-based and reference-driven. Surfaces must point to their zone and to their construction; constructions must point to their materials; transparent elements must remain linked to the relevant host or bounding element; and zone-level control data must remain attached to the corresponding thermal zone. The helper does not create these links as a secondary post-processing step. They are embedded in the mapping rules from the outset. The result is not a flat JSON export of IFC-derived values, but a structured set of interrelated JSON objects that reproduces, in the form required by the calculation engine, the spatial, constructive, and analytical relations encoded in the validated IFC model.

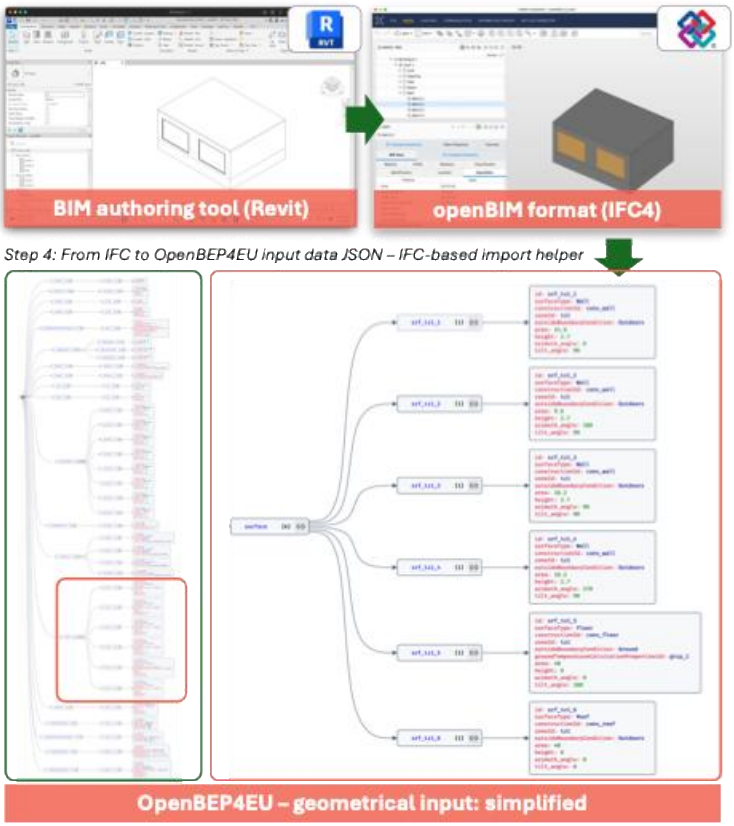


Figure 7 – Rule-based transformation of the validated IFC model into the OpenBEP4EU JSON data model, illustrated for the simplified geometrical input

As a continuation of the demonstrator workflow presented in Steps 2 and 3, Figure 7 shows how the same IFC model, once exported and validated, is transformed into the corresponding JSON structure required by the calculation engine. The upper part of the figure shows the progression from the BIM authoring environment in Revit to the exported IFC model, which serves as the validated source for the mapping process. The lower part visualises how the import helper applies a subset of its rule base to interpret IFC content and generate

the corresponding JSON object structure. In the example shown, the emphasis is placed on the simplified geometrical input, where IFC space-boundary relations, together with the associated building elements and their properties, are transformed into JSON surface objects with explicit references to zones, constructions, and boundary conditions. The figure illustrates the logic of the transformation process through a representative subset focused on geometrical and boundary-related objects.

3.3 Data validation

In the OpenBEP4EU Calculation Engine, validation is not limited to generic schema parsing. A dedicated validation module has been developed to combine standard JSON Schema validation with additional checks on references and semantic constraints. The purpose of validation is to ensure that a model is not only structurally correct, but also logically coherent and semantically valid before calculations begin. The examples that follow are simplified on purpose. They are not intended to represent complete project models, but compact cases that isolate specific validation behaviours. Even in this reduced form, they show how the validation process operates on actual object structures rather than on abstract rules alone.

The first example, illustrated in Listing 12, concerns referential integrity. The surface `wall_1` includes a `zoneId` with the value `zone_X`, yet no such object exists in the `zone` collection. The example is useful precisely because the rest of the object remains plausible: the surface has an identifier, a valid construction reference, and a valid simplified geometry description, while the construction itself is supported by an opaque material definition. The failure isolates the specific issue under examination, namely an unresolved reference. The resulting error message identifies both the exact property path and the missing target collection, which is essential for debugging imported or automatically generated data.

```
{
  "surface": {
    "wall_1": {
      "id": "wall_1",
      "zoneId": "zone_X",
      "constructionId": "wall_constr",
      "outsideBoundaryCondition": "External",
      "area": 10.0,
      "height": 2.5,
      "azimuth_angle": 0,
      "tilt_angle": 90
    }
  },
  "construction": {
    "wall_constr": {
      "id": "wall_constr",
      "constructionClass": "mass equally distributed",
      "outsideLayer": "mat_1"
    }
  },
  "materialOpaque": {
    "mat_1": {
      "id": "mat_1",
      "roughness": "MediumSmooth",
      "thickness": 0.20,
      "conductivity": 1.00,
      "density": 2000.0,
      "specificHeat": 1000.0
    }
  }
```

```
}
```

```
>> surface.wall_1.zoneId: 'zone_X' not found in 'zone'
```

Listing 12 – Example of invalid reference to a non-existing object

The second validation example, shown in Listing 13, concerns semantic correctness rather than referential existence. In this case, the reference to `zone_1` can be resolved, but the referenced object is typed as a `lighting service area` rather than as a `thermal zone`. The distinction matters. From a structural perspective, the identifier exists and the object is reachable. From a modelling perspective, however, the relationship is invalid, since the surface is expected to belong to a thermal calculation context. This example demonstrates why reference checking in the OpenBEP4EU model is not limited to existence alone: it also preserves the intended meaning of the relationships within the data model.

```
{
  "zone": {
    "zone_1": {
      "id": "zone_1",
      "type": "lighting service area",
      "volume": 100.0,
      "floorArea": 40.0
    }
  },
  "construction": {
    "wall_constr": {
      "id": "wall_constr",
      "constructionClass": "mass equally distributed",
      "outsideLayer": "mat_1"
    }
  },
  "materialOpaque": {
    "mat_1": {
      "id": "mat_1",
      "roughness": "MediumSmooth",
      "thickness": 0.20,
      "conductivity": 1.00,
      "density": 2000.0,
      "specificHeat": 1000.0
    }
  },
  "surface": {
    "wall_1": {
      "id": "wall_1",
      "zoneId": "zone_1",
      "constructionId": "wall_constr",
      "outsideBoundaryCondition": "External",
      "area": 10.0,
      "height": 2.5,
      "azimuth_angle": 0,
      "tilt_angle": 90
    }
  }
}
```

```
>> surface.wall_1.zoneId: referenced zone must have type=='thermal zone'
```

Listing 13 – Example of a reference which exists but violates semantic constraint

A third validation example is illustrated in Listing 14, which concerns logical completeness. The thermostat object `t1` is present and includes a valid cooling control through `coolingSetpointSchedule`, which itself resolves correctly to a constant schedule object. What is missing is the heating-side definition. The example does not show an incomplete object in a generic sense, but a very specific failure to satisfy the control logic formalised in the schema. The resulting error message reflects that logic directly by stating that either `heatingSetpointSchedule` or `constantHeatingSetpoint` must be provided. This is an example of validation operating at the level of modelling intent rather than simple field counting.

```
{
  "scheduleConstant": {
    "cool_sched": {
      "id": "cool_sched",
      "value": 26.0
    }
  },
  "thermostat": {
    "t1": {
      "id": "t1",
      "coolingSetpointSchedule": "cool_sched"
    }
  }
}
```

```
>> thermostat.t1: must define either 'heatingSetpointSchedule' or 'constantHeatingSetpoint'
```

Listing 14 – Example of a thermostat definition missing required control information

The example in Listing 15 addresses consistency between alternative modelling approaches. The surface `wall_1` is defined using the simplified geometry route, through area, height, azimuth, and tilt, but it also includes a `vertices` array. Since the schema defines these two representations as mutually exclusive, the object becomes invalid. The point of the example is not merely to show that two fields conflict. It shows that the validation process protects the model from mixed representations that would make downstream interpretation ambiguous. This prevents the engine, or any connected transformation service, from having to guess which geometry definition should take precedence.

```
{
  "zone": {
    "zone_1": {
      "id": "zone_1",
      "type": "thermal zone",
      "volume": 100.0,
      "floorArea": 40.0
    }
  },
  "construction": {
    "wall_constr": {
      "id": "wall_constr",
      "constructionClass": "mass equally distributed",
      "outsideLayer": "mat_1"
    }
  },
  "materialOpaque": {
    "mat_1": {
      "id": "mat_1",
      "roughness": "MediumSmooth",
      "thickness": 0.20,
      "conductivity": 1.00,
```

```
      "density": 2000.0,
      "specificHeat": 1000.0
    }
  },
  "surface": {
    "wall_1": {
      "id": "wall_1",
      "zoneId": "zone_1",
      "constructionId": "wall_constr",
      "outsideBoundaryCondition": "External",
      "area": 10.0,
      "height": 2.5,
      "azimuth_angle": 0,
      "tilt_angle": 90,
      "vertices": [
        [0.0, 0.0, 0.0],[4.0, 0.0, 0.0],[4.0, 2.5, 0.0],[0.0, 2.5, 0.0]]
    }
  }
}
```

```
>> surface.wall_1: must match exactly one geometry definition (simplified or
detailed)
```

Listing 15 – Example of conflicting geometry representation

The final example, given in Listing 16, concerns schedule linkage. The thermostat `t1` includes a valid cooling schedule reference, but its `heatingSetpointSchedule` points to `missing_sched`, which cannot be found in either of the permitted schedule collections. The significance of this case lies in the role schedules play in the calculation engine. They are not passive metadata, but active drivers of system behaviour over time. A broken schedule reference therefore affects more than the internal consistency of the file; it compromises the interpretability of the control definition. The validation message reflects this by stating not only that the target is missing, but also the specific collections in which it was expected to be found.

```
{
  "scheduleConstant": {
    "cool_sched": {
      "id": "cool_sched",
      "value": 26.0
    }
  },
  "thermostat": {
    "t1": {
      "id": "t1",
      "heatingSetpointSchedule": "missing_sched",
      "coolingSetpointSchedule": "cool_sched"
    }
  }
}
```

```
>> thermostat.t1.heatingSetpointSchedule: 'missing_sched' not found in
['scheduleConstant','scheduleCompact']
```

Listing 16 – Example of invalid schedule reference

3.4 Data transformation

Once the input data have been imported and validated, they must be converted into the specific input structures required by the core simulation modules. This is the role of the data transformation layer. Its purpose is to bridge the gap between the canonical OpenBEP4EU JSON data model, which provides a general and object-

based representation of the building and its systems, and the calculation-oriented inputs expected by the individual core modules. This step is necessary since the JSON data model is intentionally generic and interoperable. It is designed to represent the building, its spaces, envelope elements, systems, schedules, and relationships in a consistent way across the engine. The core modules, however, do not operate directly on this full representation. Each module requires a more focused set of inputs, expressed in the form most appropriate for its own calculation logic. In practice, this means that the validated JSON model must be decomposed, filtered, reorganised, and translated into structured input objects, variables, and dictionaries before calculation can begin.

For this reason, the OpenBEP4EU Calculation Engine includes a set of transformation helpers, with one helper associated with each core simulation module or module family. These helpers are responsible for extracting from the validated JSON model only the information needed by the corresponding module, converting it into the required computational form, and ensuring that the resulting inputs are complete and internally consistent. In this sense, the helpers are not merely format-conversion utilities. They implement the mapping from the shared engine-wide data representation to the module-specific representation required by the calculation routines. The transformed outputs may take different forms depending on the needs of the receiving module. In some cases, the helper produces structured Python objects, for instance to represent thermal zones, boundaries, constructions, schedules, or technical-system components in a form directly usable by a given module. In other cases, it produces dictionaries, arrays, or individual variables that gather the numerical quantities needed for specific calculations. The transformation is always performed with the target module in mind. Each helper prepares exactly the subset and structure of information that the downstream calculation routine expects to receive.

This layer also plays a key role in supporting the modular architecture of the engine. Since the transformation process is located in dedicated helpers rather than embedded directly inside the simulation modules, the core modules can remain focused on the calculations. In other words, the modules are relieved from having to interpret the full JSON structure or navigate the complete object graph of the imported model. This separation of concerns improves maintainability and makes the system easier to extend. New modules can be introduced along with new transformation helpers without requiring the internal redesign of the entire engine.

Another characteristic of the transformation layer is that module inputs may also derive from the outputs of other modules. The engine is not organised as a set of isolated calculations, but as a calculation chain in which upstream results are often required as inputs to downstream modules. The data transformation layer therefore does not only extract information from the validated JSON model; it also prepares and passes forward intermediate outputs produced by other calculation modules whenever the workflow requires it. This is important in a standards-based engine such as OpenBEP4EU, where one standard often produces quantities that are subsequently consumed by another. The helpers thus support both the initial preparation of inputs from the JSON model and the structured handover of calculation results across modules.

Special attention has been given to the terminology and naming conventions used in this layer. As far as possible, the nomenclature of inputs and outputs follows the conventions used in the relevant standards. This applies both to the naming of variables and to the way quantities are grouped and described. The intention is to preserve traceability between the implementation and the normative source, so that a variable handled in the software can be related back to the corresponding variable in the standard more easily. This same principle has guided the preparation of ANNEX A, which complements the present section by documenting the parameters identified during the analysis and implementation of the ISO 52000 family and associated standards. The relationship between the transformation layer and ANNEX A is particularly important. The helpers operate on concrete computational inputs, but these inputs are not defined in an arbitrary way. They are derived from the parameter mapping established during the standards analysis and documented in ANNEX A. In that sense, ANNEX A provides the reference vocabulary and variable mapping, while the transformation helpers provide the logic that converts the validated JSON model into the corresponding module-level inputs Top of Form.Bottom of Form.

3.5 Building performance simulation: core modules

The core of the OpenBEP4EU Calculation Engine is organised as a Python library structured by calculation domain. This constitutes the main reusable software contribution of the engine and the part intended for public release through the following GitHub repository: Que-Technologies/openbep4eu-calculation-engine. Rather than concentrating the implementation in a single monolithic solver, the code base is divided into domain-oriented modules that reflect the internal logic of the EPB calculation chain and support independent development, testing, and extension of the different parts of the framework.

`openbep4eu` At top level, the repository is organised under the `openbep4eu` package namespace. Within this namespace, the core modules are grouped into subpackages that mirror the principal domains of the EPB framework. These include `building_as_such`, `heating_systems`, `cooling_systems`, `ventilation_systems`, `domestic_hot_water_systems`, `lighting`, `building_automation_and_control`, and `renewables`. This grouping is not merely a matter of code organisation. It is the software expression of the modular architecture described earlier in the deliverable, in which each major calculation domain is treated as a distinct but interoperable component within a common framework.

`building_as_such` The `building_as_such` package occupies a central position in this structure, since it implements the core physical modelling of the building itself. It comprises a large number of modules, expected to be on the order of forty to fifty, which together calculate the time-dependent thermal behaviour of the building at zone level by combining physical building properties, climatic data, boundary conditions, and operational profiles. Its role is to provide the demand-side calculations on which the downstream technical-system modules depend. Within this package, the implemented modules cover the preparatory and computational steps required to assemble and solve the thermal behaviour of the building. Part of the package is dedicated to collecting and preparing the required inputs, including climatic information from EPW files or dedicated APIs, together with the building configuration data needed for subsequent calculations. Other modules define how thermal zones and their associated elements are represented computationally, including the discretisation of building elements into layers and nodes where the thermal calculations take place. In this context, the typology of the element, for example whether it is opaque or transparent, influences how it is represented internally and how many calculation nodes are introduced into the zone model. A substantial part of `building_as_such` is devoted to deriving the thermodynamic quantities needed by the thermal model. This includes conductance, areal heat capacity, and the convective and radiative heat-transfer coefficients that govern the exchange of heat between surfaces and their surroundings. These quantities are fundamental to the hourly energy balance, since they determine how heat is transmitted, stored, and exchanged within the thermal zone. The package also includes modules that calculate solar, internal, and device-related gains, as well as the principal heat-loss mechanisms associated with the envelope, thermal bridges, and radiative exchange. Ventilation and airflow effects are represented through routines that quantify airflow rates and their associated thermal effects, while building operation is represented through modules covering occupancy patterns, setpoint and setback temperatures, and related operational constraints. Ground interaction is also addressed explicitly through dedicated routines treating the thermal exchange between the ground and the affected building elements for cases such as slabs on ground, suspended floors, or basement-related configurations. All these quantities are then assembled in the main energy-balance routines that determine the relevant zone temperatures and the hourly sensible heating and cooling needs. In parallel, separate routines handle the latent-load calculations associated with humidification and dehumidification, and dedicated modules derive hourly domestic hot water needs from inputs related to building typology, topology, and operation profiles. This internal structure shows that `building_as_such` is not a single calculation routine, but a package of tightly related modules that represent the physical and operational behaviour of the building.

A representative example of the internal logic of the `building_as_such` package is provided by the module `thermal_needs.py` (see Listing 17), which acts as one of the main execution points of the building-side simulation workflow. In practical terms, this module organises the hourly thermal-needs simulation at zone

level by coordinating the prepared thermal-zone contexts, the climatic data, and the underlying stepwise zone-balance routine. Its role is not to define the full physical model by itself, but to drive the execution of that model over the simulation period and to assemble the resulting outputs in a structured form. The implementation reflects this role clearly. The main routine, `run_energy_simulation(...)`, receives as inputs the number of thermal zones, their identifiers, a dictionary of preconfigured `ThermalZoneContext` objects, and the relevant climatic data packaged as `EPWData`. Before the time-stepping begins, the routine initialises each zone context and builds the corresponding static matrices, thereby preparing the internal state required by the subsequent hourly calculations. It then iterates over the simulation timeline, including a warm-up period derived from the last month of the weather dataset, and for each time step calls the lower-level `step_zone(...)` routine for every zone. At each step, the module collects the principal simulation results, namely the active heating or cooling need, the indoor air temperature, and the operative temperature. These quantities are stored over time and, once the warm-up period has been discarded, are assembled into an `EnergySimulationResult` object. In this way, the module provides a clear example of how the OpenBEP4EU Calculation Engine combines typed input structures, time-dependent climatic data, zone-level state handling, and repeated execution of a standards-based balance routine in order to produce structured simulation outputs.

```
from __future__ import annotations
import numpy as np
from openbep4eu.building_as_such.models.zone_context import ThermalZoneContext
from openbep4eu.building as such.models.step zone import step zone
from openbep4eu.building_as_such.models.types import EnergySimulationResult
from openbep4eu.building_as_such.models.epw import EPWData

def run_energy_simulation(
        n_zones: int,
        zone ids: list[str],
        contexts: dict[str, ThermalZoneContext],
        epw_data: EPWData,) -> tuple[EnergySimulationResult, dict]:

    for ctx in contexts.values():
        ctx.init state()
        ctx.build_static_matrix()

    timeline = epw_data.data.index
    last_month = epw_data.data.index.month[-1]
    warmup_timesteps = int((epw_data.data.index.month == last_month).sum())
    n timesteps      = len(timeline) + warmup timesteps
    Phi_HC   = []
    Theta_int = []
    Theta_op  = []

    for t in range(n_timesteps):
        phi t   = np.zeros(n zones, dtype=float)
        t_int_t = np.zeros(n_zones, dtype=float)
        t_op_t  = np.zeros(n_zones, dtype=float)

        for m, zid in enumerate(zone_ids):
            ctx = contexts[zid]
            ctx_t = t
            result = step_zone(ctx=ctx, t=t)
            phi_t[m]   = result.phi_hc_nd_act
            t_int_t[m] = result.theta_int_act
            t_op_t[m]  = result.theta_op_act

        Phi_HC.append(phi_t)
        Theta_int.append(t_int_t)
        Theta_op.append(t_op_t)


    return EnergySimulationResult(
        timeline=epw_data.data.index,
        Phi_HC=np.array(Phi_HC[warmup_timesteps:]),
        Theta_int=np.array(Theta_int[warmup_timesteps:]),
        Theta_op=np.array(Theta_op[warmup_timesteps:]),), contexts
```

Listing 17 – OpenBEP4EU Calculation Engine's core modules: thermal_needs.py

This example is representative of the broader design approach of the core modules. The modules do not operate as isolated scripts; they are built around typed domain objects, explicit execution routines, and structured result containers. In the case of `thermal_needs.py`, this design supports the hourly simulation of multiple thermal zones while keeping the execution logic, the underlying physical calculations, and the resulting outputs clearly separated and reusable within the wider engine.

`technical_systems` The same domain-based logic extends to the `technical_systems` modules, grouped into `heating_systems`, `cooling_systems`, `ventilation_systems`, `domestic_hot_water_systems`, `lighting`, `building_automation_and_control`, and `renewables`. The OpenBEP4EU Calculation Engine has been designed to include dedicated components that model the different stages of the technical-system calculation flow and translate building needs into actual system-related energy use. These modules do not represent technical systems as monolithic objects. Instead, they are grouped according to the main functional segments of integrated building energy systems and allow these segments to be combined in a modular and scalable manner. In this structure, the implemented system modules distinguish between emission, distribution, storage, and generation functions. Emission-related modules address the building components that deliver heating or cooling to the conditioned spaces by transferring energy to the room air and surrounding surfaces. Distribution-related modules address the transport of heating or cooling energy from the generation level to the relevant end-use components. Storage-related modules address the temporary retention of thermal energy so that it can be used when required. Generation-related modules address the equipment that produces the heating or cooling energy supplied to the building. This decomposition is important because it allows the engine to represent realistic technical systems as combinations of interacting components rather than as simplified single devices. The design of the OpenBEP4EU Calculation Engine enables these different system segments to be modelled both individually and in combination. This is a key architectural feature, since actual technical building systems are inherently composite. Hence, the implemented structure is granular and extensible: numerous dedicated sub-modules describe the behaviour of the different segments of each supported system, making it possible to represent a broad range of configurations while preserving modularity and reusability. As of M18 of the OpenBEP4EU project, the implemented scope already covers a representative range of technical building systems. On the emission side, this includes different types of heating and cooling emitters, such as radiators, panel heaters, floor-heating systems including TABS, radiant ceiling panels, luminous heaters, radiant tube heaters, and back-up heaters. On the generation side, the current implementation includes combustion-based systems, such as gas, oil, solid-fuel, biomass, and multi-fuel boilers, together with several heat-pump typologies, including AC split units, air-to-air heat pumps, VRV systems, air-to-water heat pumps, and ground-to-water heat pumps. Cooling-related modules extend this structure to air-cooled and water-cooled chillers as well as heat-rejection systems. The ventilation domain covers mechanical ventilation components such as air-handling units and heat exchangers. The system coverage also extends to storage and domestic hot water, including thermal storage for heating and domestic hot water, electric water heaters, circulation water heaters, and solar water heaters. In addition, the current implementation includes further energy-related domains such as photovoltaic generators, district heating and district cooling systems, and lighting.

`pyproject.toml` and `requirements.txt` At repository level, this package structure is complemented by `pyproject.toml`, which defines how the released core is managed as a Python project. This file contains the project metadata, package definition, and dependency declarations required to install and use the calculation engine as a Python package. Its practical role is to turn the repository into an installable software library rather than a loose collection of source files. In the context of the public release, this is particularly important, since it allows users not only to inspect the source code, but also to install the package from the project root in a controlled Python environment and import the exposed modules under the `openbep4eu` namespace in a consistent way. In this sense, `pyproject.toml` provides the package-level installation and build logic of the released core, while the accompanying `requirements.txt` file records the dependency environment associated with the released version.

3.6 Building performance simulation: orchestration

The orchestration layer is responsible for the execution of the OpenBEP4EU Calculation Engine as a complete simulation workflow. While the core modules implement the standards-based calculations for specific domains, orchestration arranges these modules into an execution sequence. Its role is to ensure that each calculation is triggered at the appropriate stage, with the correct inputs, and in a way that preserves the dependencies that exist across the EPB calculation chain.

In practical terms, the orchestration layer is implemented in Python through workflow-control routines operating above the core modules and their associated transformation helpers. Starting from the validated JSON model, the orchestration process establishes the simulation context, determines which modules are required for the case under consideration, and invokes the relevant helpers to prepare the corresponding inputs. These transformed inputs are then passed to the appropriate module functions or classes, and the resulting outputs are retained in structured form so that they can either feed subsequent calculations or be exposed as final engine outputs. Thus, orchestration is not a computational module. It is the control layer that coordinates the movement of Python objects, dictionaries, typed structures, and time-dependent arrays through the calculation chain. In a typical execution path, it first prepares the building-side inputs, then triggers the relevant building-side calculations, then passes the resulting quantities to the appropriate technical-system modules, and finally assembles the outputs required by the rest of the engine. The same principle applies when the workflow includes optional modules, alternative execution paths, or cases in which one module depends partly on the results produced by an earlier stage.

A core responsibility of this layer is the management of intermediate results. In a standards-based engine, many quantities are not final outputs but are required by downstream calculations. For this reason, orchestration maintains structured containers for the outputs produced at each stage, so that they can be passed forward in a controlled and traceable way rather than recomputed or repeatedly extracted from the original data model. This is important in hourly calculations, where climatic data, schedules, zone-level temperatures, loads, and other state-dependent variables must remain consistent across the full simulation chain. The same logic applies to the distinction between intermediate quantities and reportable outputs. Some results are needed only internally, in order to support later steps of the workflow, whereas others constitute the outputs that must ultimately be made available to reporting, API services, or other uses. The orchestration layer manages both categories explicitly: it preserves internal quantities for continuity of the calculation chain and, at the same time, collects and organises the results that form the external output of the engine. Another key role of orchestration is to preserve the overall simulation context throughout execution. This includes maintaining the correct associations between building entities, technical systems, schedules, operational assumptions, and time-dependent variables as the workflow progresses. In implementation terms, this means that orchestration does not simply trigger calculations one after another; it ensures that each module is executed against a consistent and traceable set of assumptions, inputs, and intermediate states.

3.7 Technology stack

The OpenBEP4EU Calculation Engine is implemented in Python and relies on a broad technology stack supporting typed data modelling, API exposure, numerical and tabular processing, weather and solar calculations, spreadsheet interoperability, and IFC/IDS-based import and validation. The technology stack described in this section covers the software dependencies used across the OpenBEP4EU Calculation Engine as a whole. This includes dependencies supporting the publicly released components, as well as dependencies used by components that remain internal to the project implementation.

The Python dependencies required for the publicly released components (i.e., the building performance simulation core modules) will be documented explicitly in a requirements.txt file, which will be made available through the aforementioned GitHub repository along with the released core modules. The public release

targets Python 3.11, and the repository documentation will state this supported version explicitly so that external users and developers can reproduce the execution environment consistently. This is particularly important in the present case, since the calculation engine depends on a combination of numerical, interoperability, and API-oriented packages whose compatibility is partly version-dependent. The main packages currently used in the calculation stack, together with their role in the OpenBEP4EU Calculation Engine and their indicative licensing scheme, are the following:

1. pydantic and typing-extensions are used to define and validate the structured Python objects that mirror the OpenBEP4EU JSON data model. They support typed representation of inputs, intermediate objects, and outputs, and contribute to consistency across the different modules of the engine.
2. fastAPI is used to expose the calculation engine through API endpoints. Within the OpenBEP4EU framework, this supports services related to validation, execution, orchestration, and access to results.
3. numpy and pandas provide the numerical and tabular backbone of the engine. They are used for hourly calculations, time-series handling, schedules, processing of weather-related datasets, intermediate result organisation, and result aggregation.
4. pvlib is used for weather and solar-related processing. In the OpenBEP4EU Calculation Engine, its role includes reading weather data in EPW format, processing the corresponding climatic series, calculating solar position and solar angles, and supporting irradiance-related calculations required by the thermal and solar parts of the workflow.
5. openpyxl is used for spreadsheet interoperability. This is relevant for handling structured spreadsheet-based material, including tabulated inputs, expert-provided calculation sheets, and output results in Excel-compatible form.
6. ifcopenshell is the main package for the IFC-based import workflow. It is used to access IFC entities, property sets, material definitions, space-boundary relationships, and other model relations needed by the IFC-based import helper. In this way, it supports the rule-based transformation of validated IFC content into the OpenBEP4EU JSON data model.
7. ifctester is used for IDS-based validation of IFC inputs. Within the engine workflow, it provides the formal validation step that checks whether an IFC file satisfies the information requirements expected by the import route before the IFC-to-JSON mapping is executed.

3.8 Installation and license

The publicly released software artefacts of the OpenBEP4EU Calculation Engine will be made available through the dedicated GitHub repository. In line with the modular architecture described earlier, this public release concerns the building performance simulation core modules, which are provided as Python source files and represent the main reusable software contribution of the engine. The surrounding layers that support orchestration, validation services, helper-based input preparation, and project-specific integration are more closely tied to the operational implementation of the engine within the OpenBEP4EU framework and are not part of the main public release.

Installation From an installation perspective, the public repository is intended to provide the entry point for users who wish to inspect, test, and reuse the released core modules. The software dependencies and technology stack required for execution are documented in the repository and listed explicitly in the requirements.txt file, as described in Section 3.7. Installation is expected to follow the standard Python workflow, namely retrieval of the repository, creation of a suitable Python environment, and installation of the required packages from the provided dependency file. In this way, the installation procedure remains directly aligned with the released implementation and can evolve together with it through version-controlled updates in the repository. This approach has two advantages. First, it makes the released modules easier to reproduce and evaluate, since the execution environment can be reconstructed directly from the repository contents. Second, it keeps

the public release technically consistent with the actual implementation, avoiding the need to duplicate dependency information in static documentation. Any module-specific usage notes, updates, or version constraints can likewise be maintained directly within the repository documentation.

Licensing Regarding licensing, the publicly available core modules are released under the Mozilla Public License 2.0 (MPL-2.0). This license supports openness, reuse, and further technical development, while establishing clear conditions for redistribution and modification of the licensed source files. The applicable license text is published in the GitHub repository along with the released source code, so that the terms of use remain transparent and directly accessible to users. For the purposes of the present deliverable, it is therefore sufficient to note that the released core modules are accompanied by explicit repository-level licensing information governing access, reuse, modification, and redistribution.

3.9 Assumptions and restrictions

During development, several challenges arose from the need to translate standards-based procedures into an operational software environment. In several cases, the standards offer alternative formulations, rely on tabulated values, or require expert interpretation before they can be implemented in a robust and reproducible way. To address such cases in a technically sound manner, the development team worked closely with EPBC experts to assess alternatives and agree on implementation choices. The resulting assumptions were introduced to operationalise the methodology, reduce ambiguous or error-prone inputs, and maintain a stable calculation workflow while preserving the technical credibility of the results.

Assumptions related to envelope and zone modelling The current version of the OpenBEP4EU Calculation Engine relies on the following main assumptions:

1. Thermal transmittance of transparent elements is introduced directly as an input parameter – For windows and other transparent elements, thermal transmittance is provided directly rather than being recomputed within the engine from lower-level product characteristics. This avoids the need to reconstruct detailed product descriptions, such as glazing composition, frame geometry, spacer characteristics, and insulation details, within the calculation workflow, while preserving the key thermal parameter needed by the simulation.
2. Thermal conductivity and thermal capacity are derived from known parameters where sufficient information is available – In many standards-based workflows, these properties may be taken from tabulated values provided in annexes. In the OpenBEP4EU Calculation Engine, when the required material and geometric information is available, they are derived from known input parameters instead. This approach was preferred in order to improve physical consistency and retain applicability across a broader range of constructions and building typologies. Tabulated values remain relevant as fallback data where detailed information is not available.
3. Unconditioned spaces remain explicitly represented in the thermal model – Where a zone is not actively heated or cooled, it is not removed from the building representation. Instead, it remains explicitly included in the thermal structure of the model, but with zero sensible load assigned throughout the hourly calculation. In this way, the zone continues to interact thermally with adjacent spaces and building elements, while no heating or cooling energy need is attributed to it.
4. Ground-contact heat transfer, including basement-related cases, is handled through an explicit zone-based boundary interpretation – The thermal interaction between the ground and the building depends on the floor arrangement, such as slab-on-ground, elevated floor, or basement-related configurations. In the present implementation, slab-on-ground cases are treated using the corresponding ground-coupling formulation. Where a basement is present, it is represented as a separate adjacent zone, generally an unconditioned one, rather than being excluded from the model. Under this interpretation, the basement slab and basement walls in contact with the ground remain part of the thermal structure as ground-contact

elements. This allows basement-related heat transfer to be incorporated within the same zone-based modelling framework while maintaining a technically consistent representation of the relevant boundary conditions.

Assumptions related to technical building systems In addition to the above, some assumptions were required for the modelling of technical systems:

1. Monthly average outdoor temperature is used in the emission calculation of certain air systems – In the implementation of the relevant emission formulas for air-heating and air-cooling systems, the direct use of hourly outdoor temperature led in some cases to numerical instability. Following discussion with the EPBC experts, the average monthly outdoor temperature was used instead in that part of the calculation. This choice was made to ensure numerical robustness while retaining a physically meaningful climatic input.
2. Missing annex B values for air-cooling emission systems are substituted using the corresponding air-heating values – In some cases, the relevant annex values are not available for the cooling-side equivalent of a system, while corresponding values exist for the heating-side case. In agreement with the EPBC experts, the heating-side values were used as a pragmatic substitute where no direct cooling-side values were provided.
3. Operational spreadsheet formulations were used where the standard was not directly programmable – For some systems, the corresponding standard was either not directly programmable in software form or did not provide a formulation that could be translated into code without further operational clarification. In such cases, the EPBC experts provided spreadsheet-based formulations that made the intended equations explicit. These formulations were then used as the basis for implementation in the OpenBEP4EU Calculation Engine, allowing additional systems to be addressed within the framework.

Current implementation boundaries Alongside the assumptions described above, the current version of the OpenBEP4EU Calculation Engine is subject to the following implementation boundaries that define its present scope:

1. The execution workflow currently relies on the OpenBEP4EU JSON data model – At present, the engine operates on structured JSON input consistent with the OpenBEP4EU data model. Although an IFC-based import route has been developed through the dedicated helper described earlier, the core execution workflow remains centred on this canonical JSON representation. In addition, highly irregular or geometrically complex building cases are not yet comprehensively addressed.
2. The framework currently requires a Python execution environment – The engine is implemented in Python and requires the prior installation of the necessary dependencies and packages. These prerequisites are described in the technology stack and installation sections. At this stage, the framework is primarily intended for technically informed users and developers working within the project or in related technical contexts.
3. The current release covers a defined subset of technical building systems – The systems currently integrated correspond to a defined implementation scope and include selected heating, cooling, and domestic hot water configurations, as described in Section 3.4.2 and reflected in the results presented in Chapter 4. This should be understood as the scope of the present release rather than as a fixed limitation of the architecture. Broader system coverage is already foreseen in the upcoming releases of the framework.
4. Some system categories are not yet included in the present release – Mechanical ventilation systems, cogeneration systems, renewable energy systems, and lighting systems are not yet integrated in the current release. Their absence reflects the present implementation sequence and release planning, not a conceptual exclusion from the overall framework.
5. A dedicated shading module has not yet been incorporated into the present release – The current version does not yet include a complete shading module (planned to be fully covered in the next release). This is expected to influence the accuracy of sensible load calculations, particularly in cases where solar gains and solar control have a significant impact on zone-level thermal behaviour.

6. The current scope is aligned with the building categories addressed in the implemented methodology – The present implementation addresses the building types covered by the project scope and by the applicable calculation framework, primarily residential buildings and small office buildings. Industrial buildings and other specialised building categories are outside the scope of the current release.

User Interface It should also be noted that user interaction and results visualisation are not handled within the calculation engine itself. These aspects are addressed through the common OpenBEP4EU user interface being developed in parallel by the project coordinator. For this reason, the absence of a dedicated standalone user interface within the engine should not be interpreted as a restriction of the calculation core, but rather as a consequence of the overall platform-level division of responsibilities.

Assumptions and limitations related to Building Automation and Control Systems (BACS) (EN ISO 52120-1)

The current release implements a subset of control-related inputs and parameters where these are explicitly required by the EPB modules already integrated in the engine (e.g., zone setpoints and schedules, and selected system control parameters where supported). Some higher-level BACS functions (e.g., supervisory optimisation, cross-system coordination, and certain interlock strategies) are not yet represented quantitatively, as the corresponding integration logic across modules and kernel identifiers has not yet been fully defined.

To ensure transparency and to avoid double counting, going forward it is recommended to maintain a BACS Function Coverage Register aligned with the EN ISO 52120-1 function identifiers. For each function, the register should indicate whether it is:

- Quantified (direct): represented through an explicit kernel parameter/identifier that affects results;
- Quantified (proxy): represented indirectly via an EPB module parameter/option (with a short explanation of what the proxy captures and what it does not);
- Not quantified / out of scope: not represented in the current release (to be treated as a documented limitation).

A short, reusable limitation statement based on this register should be included in user documentation and relevant deliverables, to make clear which control functions are currently captured and which are not.

4 Test cases and preliminary results

This section presents the test cases used to evaluate the current implementation of the OpenBEP4EU Calculation Engine and the corresponding preliminary results. The purpose of this exercise was to assess whether the implemented modules produce numerically consistent results under a set of controlled scenarios, and to compare these results with recognised reference material and with a widely used building simulation environment. To achieve this, the assessment followed two complementary approaches.

The first approach consisted of reference-based testing against EPB spreadsheets. For the implemented calculation modules, the corresponding formulas were tested against official or expert-provided EPB spreadsheet material. In practice, the same input values were applied both in the OpenBEP4EU Calculation Engine and in the corresponding spreadsheet, and the resulting outputs were compared. The objective of this approach was not to claim formal validation of the whole framework, but to check whether the implemented formulas behave consistently with the corresponding EPB calculation procedures and produce results of the expected order and structure. Since the engine has been developed as a modular Python framework, this type of testing was carried out at module level. The second approach consisted of a comparative assessment against EnergyPlus software. The same set of representative test cases was modelled in EnergyPlus and the resulting outputs were compared with those generated by the OpenBEP4EU Calculation Engine. The purpose of this comparison was not to demonstrate exact numerical agreement between the two engines. Such an outcome would not be expected, since the two frameworks are based on different methodological foundations: the OpenBEP4EU Calculation Engine follows the EPB standards framework, while EnergyPlus is based on a different modelling and simulation architecture. The comparison was used as a benchmarking exercise, with emphasis on the consistency of result magnitudes, trends, and temporal behaviour rather than on one-to-one numerical equivalence.

In a nutshell, the main sources of deviation between the two frameworks are associated with differences in both input structure and underlying modelling assumptions. First, the two tools do not require exactly the same inputs and do not organise physical quantities in the same way, which limits the possibility of achieving a fully equivalent input model. Second, EnergyPlus is a highly flexible and complex simulation environment whose results depend strongly on modelling choices and on the experience of the modeller. Relevant examples include the treatment of ground temperature, which in the OpenBEP4EU Calculation Engine is derived according to ISO 13370 but in EnergyPlus is introduced externally; the handling of shading, which is represented in EnergyPlus but is not yet included in the current release of the OpenBEP4EU Calculation Engine; and the calculation of solar irradiance, where different formulations may be applied. These methodological differences must be considered when interpreting the comparative results. Even with these differences, the comparative exercise remains informative. It provides an additional point of reference and helps assess whether the OpenBEP4EU Calculation Engine produces technically plausible results across a range of scenarios, including cases that are also simulated in a recognised building-performance tool.

To ensure a reproducible basis for these tests and comparisons, the test cases were derived from the BESTEST benchmark. BESTEST provides a standardised and internationally recognised family of reference cases with clearly defined inputs and has therefore been used as the common basis for the present assessment. In particular, the scenarios presented in this section build on the BESTEST 600 series, which represents a lightweight single-zone building with controlled internal conditions. For the purposes of the OpenBEP4EU work, these standard cases were adapted in a controlled manner to reflect the requirements of the implemented EPB-based workflow. The baseline configuration consisted of a single thermal zone derived from the BESTEST 600FF case, initially without windows, with a slab-on-ground construction, and located in Athens. The building was treated as a residential dwelling in accordance with the building typologies of EN 16798-1:2019 and assumed to be continuously occupied. Some adjustments to the original BESTEST setup were necessary, since the OpenBEP4EU Calculation Engine is not limited to the ISO 52000 family alone but also incorporates additional related standards, including EN ISO 6946 for thermal resistance and transmittance, EN ISO 13370

for ground interaction, and EN 16798-1 for operational assumptions and usage profiles. The resulting input configuration used for the modified BESTEST case is summarised in Table 2.

Table 2 – Input parameters of the modified BESTEST 600 case

Parameter	Symbol	Variable in EPB engine	Unit	Value
building type	BLDNG-CAT_TYPE	building_type	-	"Retired house apartment"
setpoints	$\theta_{int;set;H;ztc}$/ $\theta_{int;set;C;ztc}$	heating_setpoint/ cooling_setpoint	°C	26 for cooling 20 for heating
Window's glazing type	N/A	glazing_type	-	Double glazing
Colour of wall elements	N/A	color_element_type	-	intermediate color
air flow rate	$q_{V,k;t}$	q_v_k_t	m^3/h	64.8
thermal bridge heat	$H_{tr;tb;zt}$	thermal_bridge_heat	W/m	0
the areal thermal capacity of air and furniture needs	$\kappa_{m;int}$	km_int_a_ztc	$J/(m^2 \cdot K)$	10000
construction class	N/A	construction_class	-	Construction with external thermal insulation (main mass component near inside surface)
Wall Construction Elements - thickness	N/A	thickness_per_layer_matrix	m	Plasterboard (0.012) / Fiberglass Quilt (0.066) / Wood Siding (0.009)
Roof Construction Element - thickness	N/A	thickness_per_layer_matrix	m	Plasterboard (0.010) / Fiberglass Quilt (0.1118)/ Roof Deck (0.019)
Floor Construction Element - thickness	N/A	thickness_per_layer_matrix	m	Timber Flooring (0.025)/ Insulation (1.003)

From this baseline, the test cases were developed progressively. The first stage considered the free-floating single-zone case. The second stage introduced fenestration and ideal heating and cooling systems to examine the calculation of sensible thermal loads in the single-zone configuration. Figure 8 illustrates this single-zone setup through a top-down view of the model together with an indicative three-dimensional representation.

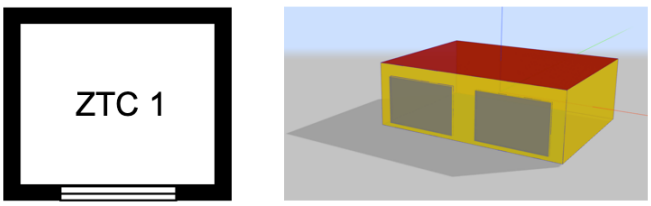


Figure 8 – Geometrical representation of the single-zone configuration used in the preliminary assessment

Building on this initial setup, the modelling approach was then extended to a multi-zone configuration by adapting the original BESTEST-based case to a more complex geometry. Additional thermal zones were introduced by replicating the initial single-zone model: one zone was positioned to the east of the original zone, while a second was placed to the north. In the latter case, the window orientation was modified to face north in order to account for differences in solar exposure. Figure 9 presents a top-down and indicative three-dimensional view of this multi-zone arrangement. This extension made allowed the development team to investigate thermal interactions between adjacent zones, as well as the effects of orientation and spatial configuration on the building's energy needs. The final stage introduced explicit technical building systems for heating, cooling, and

domestic hot water, enabling the framework to move from needs-side calculations to system-related energy use across the expanded building geometry.

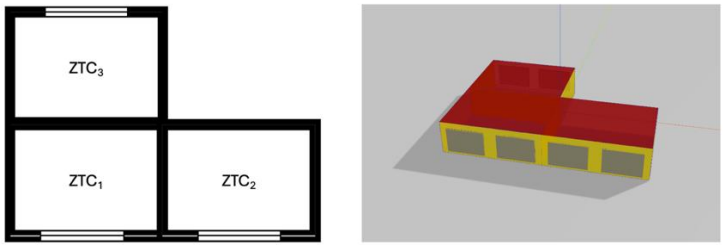


Figure 9 – Geometrical representation of the multi-zone configuration used in the preliminary assessment

The following subsections present the results in this order, starting from the free-floating single-zone case and progressively moving towards more complex configurations with ideal systems, multiple interacting thermal zones, and dedicated technical building systems.

4.1 Free-floating single-zone thermal response

In the free-floating conditions case, the focus was gathered on illustrating the thermal response of the building, with and without the inclusion of fenestration. To achieve this, four distinct days representing each season have been selected and the respective internal and operative temperature daily profiles were examined. Those temperature profiles are visualised along with the external temperature which controls heat exchange with the environment and the global horizontal irradiance (GHI) which defines the solar gains of a thermal zone.

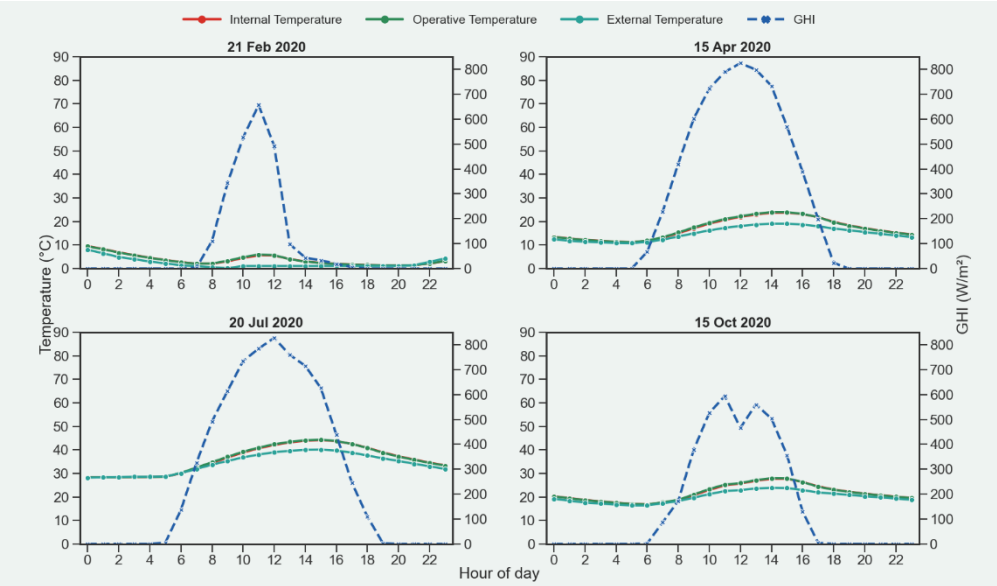


Figure 10 – Internal and operative temperatures for the free-floating single-zone test case without window

As observed in Figure 10, the indoor and operative temperatures remain relatively stable throughout the day, showing limited variation despite fluctuations in external temperature and solar irradiance. This behaviour is expected, as the absence of windows eliminates direct solar gains resulting in a thermal response driven exclusively by other factors such as the transmission through the opaque envelope, the internal heat gains and more. In contrast, when the south-oriented window is added, a significant increase in indoor and operative

temperatures is observed during daytime hours across all examined periods (Figure 11). This increase correlates strongly with peaks in global horizontal irradiance (GHI), highlighting the impact of solar heat gains on the zone's thermal behaviour. The effect is particularly more intense during spring and summer conditions, where higher solar radiation leads to elevated peak temperatures and increased daily temperature swings.

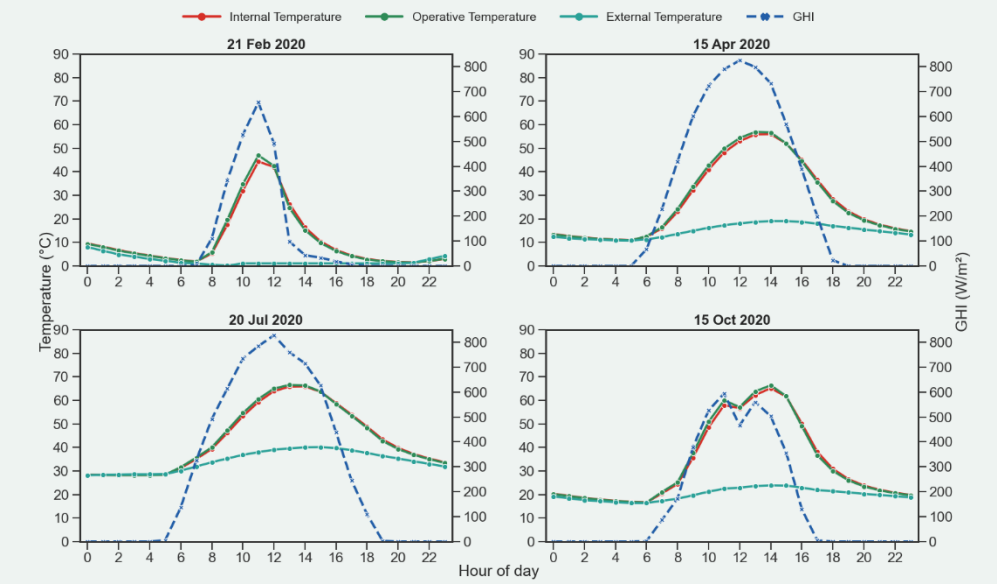


Figure 11 – Internal and operative temperatures for the free-floating single-zone test case with window

4.2 Single-zone thermal energy needs

Following the free-floating simulation, the same building configuration and envelope characteristics were retained, while ideal heating and cooling loads were introduced.

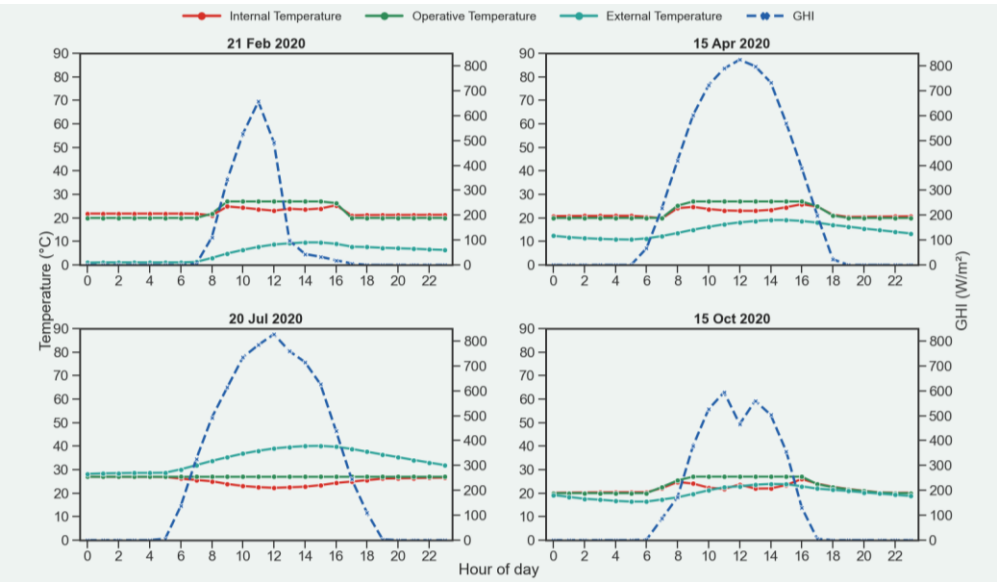


Figure 12 – Internal and operative temperatures for the single-zone test case (conditioned)

The respective heating and cooling energy needs were determined based on the energy required to maintain the indoor operative temperature within the defined setpoints. In this context, the ideal heating and cooling system operates by instantaneously supplying or removing the necessary energy at each timestep to ensure that the operative temperature does not fall below the heating setpoint (20°C) or exceed the cooling setpoint (26°C). Figure 12 presents the temperature profiles in the single-zone case after the introduction of the ideal systems. It is clearly observed that the operative temperature does not violate the setpoints in any of the examined days of each season. This behaviour is expected since the implemented capacity limits of ±10,000 W are sufficient to regulate the indoor conditions of the examined thermal zone.

4.2.1 Comparison with EPB reference spreadsheets

The single-zone simulation equipped with ideal heating/cooling system constituted the first fully validated case against the corresponding EPB spreadsheet tool, ensuring exact alignment of the calculated energy needs. Figure 13 highlights the proximity of the generated results from the two tools. In the heating loads case, the average monthly deviation was measured at **0.42%**, while in the cooling case, **0.10%**. Such deviations are considered acceptable, due to different rounding approaches implemented in python and excel.

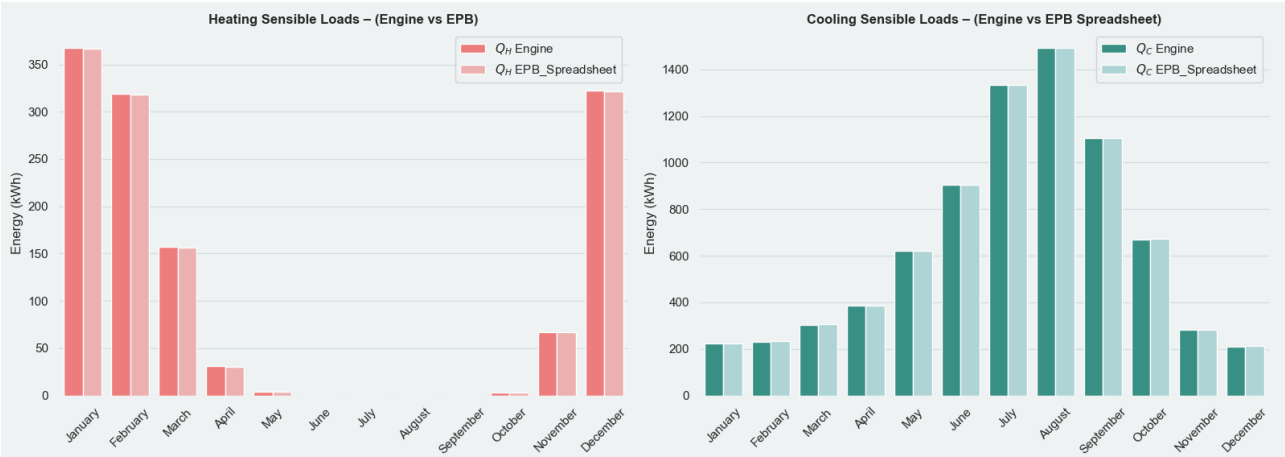


Figure 13 – Single-zone test case, Heating and Cooling needs: Comparison with EPB spreadsheets

4.2.2 Comparison with EnergyPlus software

In parallel, the same configuration was simulated in EnergyPlus, forming the initial basis for the comparative assessment between the two modelling approaches (Figure 14).

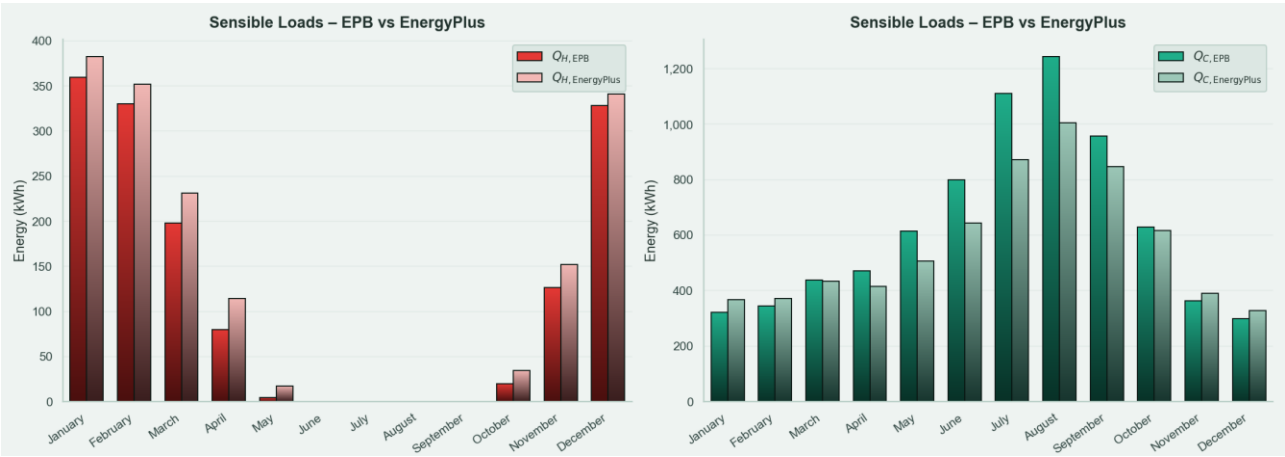


Figure 14 – Single-zone test case, Heating and Cooling needs: Comparison with EnergyPlus

The openBEP4EUCalculation Engine and EnergyPlus software results illustrate the same pattern and order of magnitude. The heating cooling needs are higher during the winter season while the cooling energy needs peak during the summer period. These results are on par with the building location (i.e., Athens) with a window faced to the south. The observed deviations (up to ±15% in heating loads and ±30% in cooling loads), however, showcase the inherent differences between the two engines.

4.3 Multi-zone thermal energy needs

Extending the analysis to a more complex geometry, allowed for a more detailed evaluation of the thermal behaviour of a building segmented into multiple thermal zones. In the multizone case, certain building elements act as partitions between adjacent thermal zones rather than being exposed to external conditions. To upscale the model to a multizone implementation, the boundary condition treatment for internal partition elements was modified. Specifically, for nodes associated with partition elements, the external temperature input was replaced with the internal air temperature of the neighbouring zone from the previous timestep.

The following figures present the internal and operative temperatures per zone, plotted against the external temperature and global horizontal irradiance (Figure 15), as well as the heating/cooling load (Figure 16), for a specific day of the year (15th of April). The operative temperature is almost identical in all three zones, mainly since it is bounded by the setpoints (i.e., 20°C and 26°C). On the contrary, the internal temperature profile appears to differ in the third zone, remaining relatively stable at approximately 25°C. This behaviour is attributed to the reduced operation of the cooling system, which ultimately has a lower impact on the internal temperature (Figure 16). In more detail, Zone 3 has fenestration with a northern orientation. In the case of Athens, this results in less solar exposure compared to the other two zones, leading to lower solar gains through the window and, consequently, reduced cooling needs

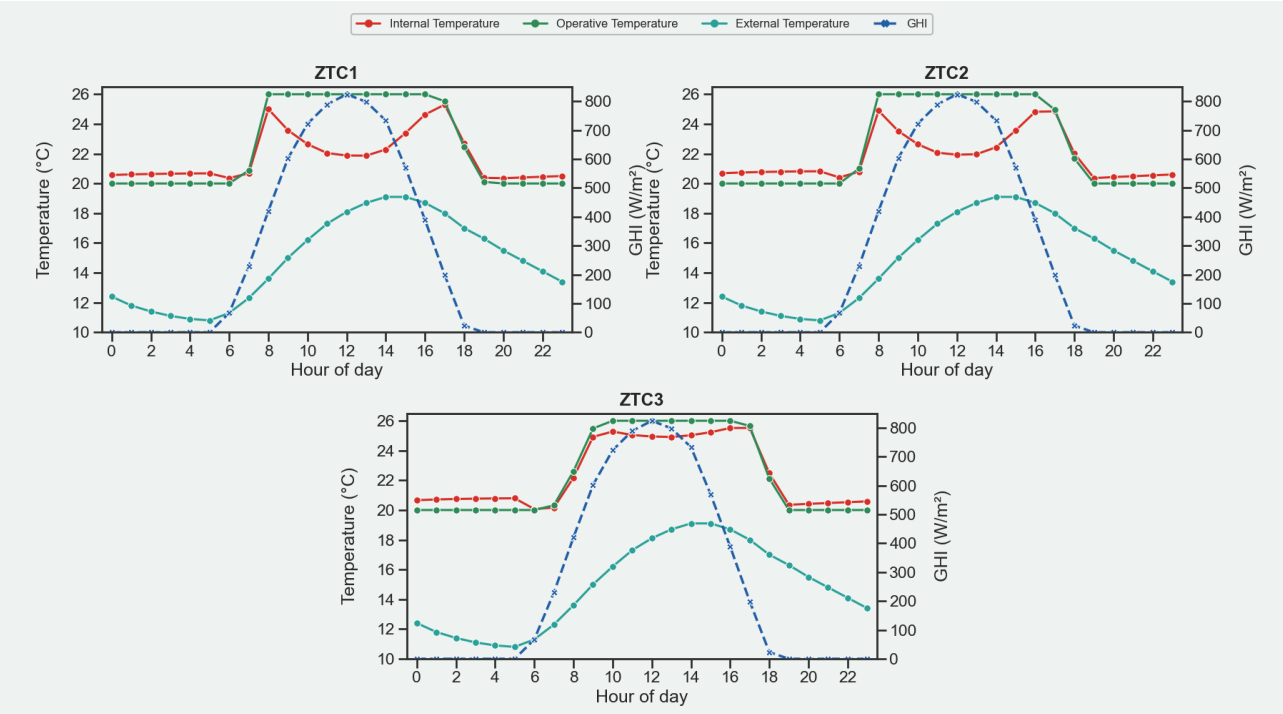


Figure 15 – Multizone internal and operative temperatures vs external temperature and GHI (15th of April)

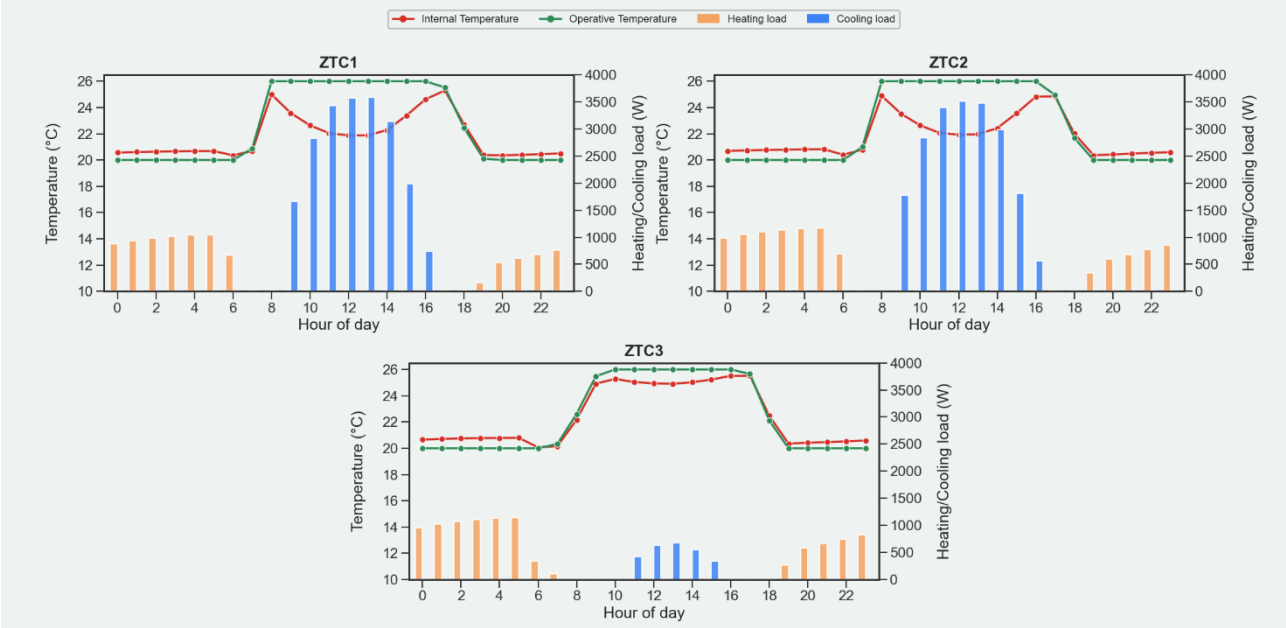
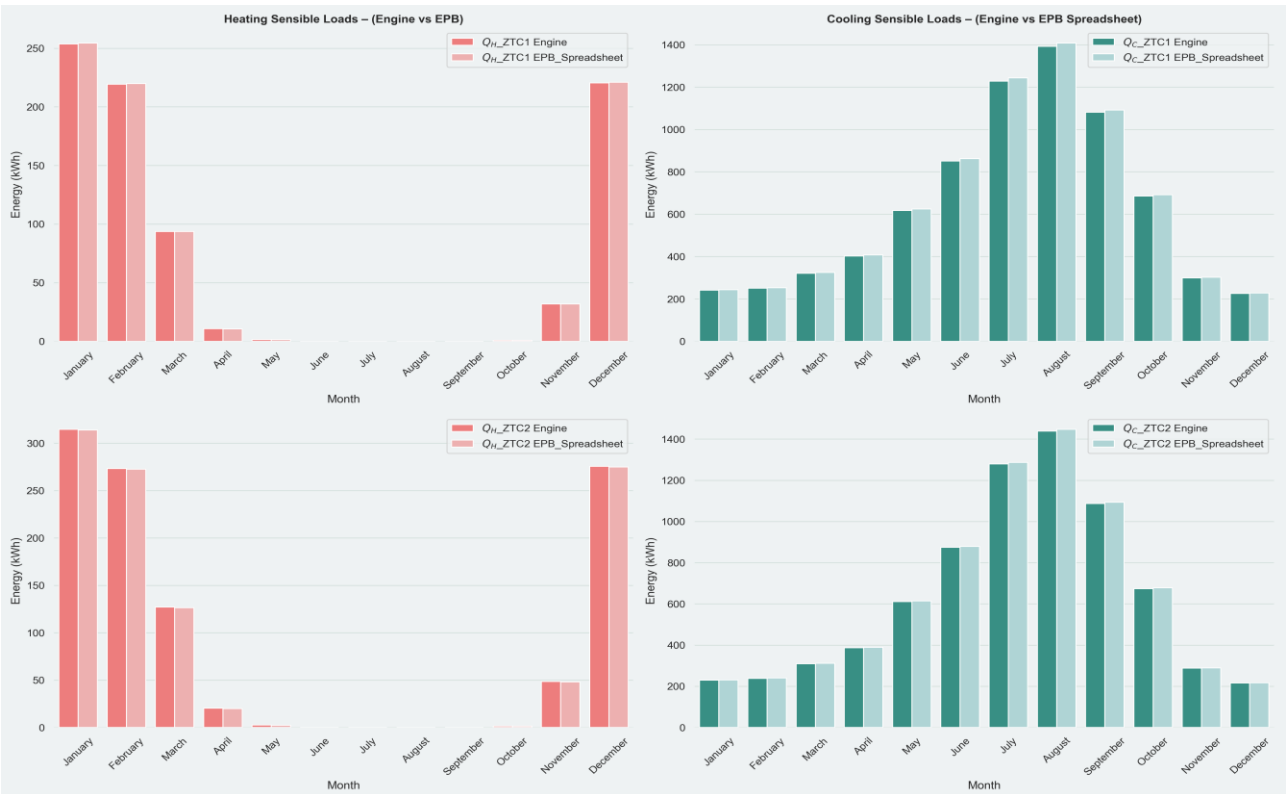


Figure 16 – Multizone internal and operative temperatures vs (ideal) Heating/Cooling load (15th of April)

4.3.1 Comparison with EPB reference spreadsheets

Similar to the single-zone case, the same validation methodology was applied to the multizone implementation. Based on identical inputs in both the engine and the EPB spreadsheet, the heating and cooling needs, as well as intermediate variables (e.g., operative/internal temperature, gains, material properties and others), were compared to ensure alignment between the two approaches.



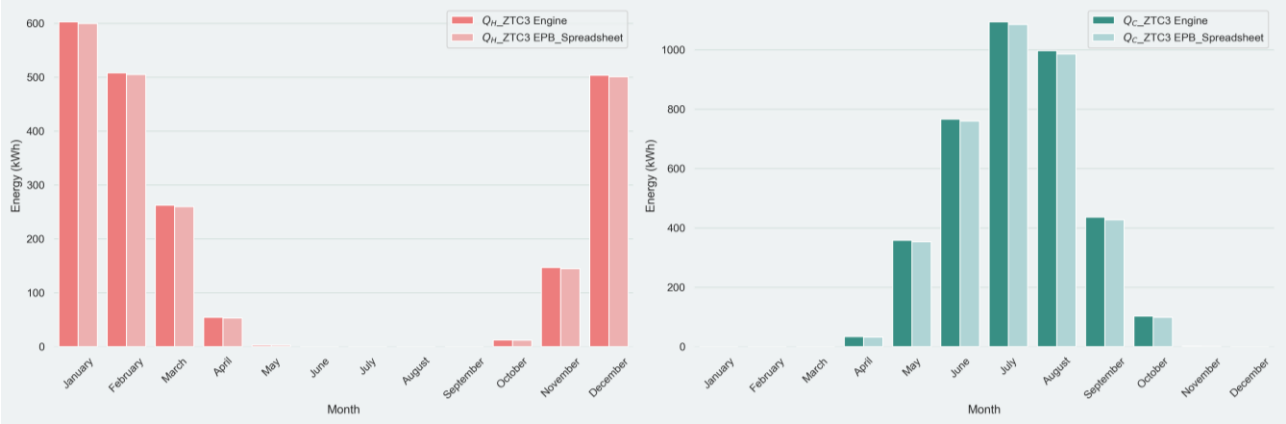


Figure 17 – Multi-zone test case, Heating and Cooling needs: Comparison with EPB spreadsheets

Figure 17 presents the results of the validation for the sensible loads case. As expected, the observed deviations were extremely low (0.27% for cooling and 0.24% for heating), again attributed to different rounding approaches.

4.3.2 Comparison with EnergyPlus software

To extend the comparative assessment to the multizone case, the single-zone EnergyPlus software input was modified to reflect the changes introduced in geometry, including zone interactions, orientation, materials and fenestration.

Figure 18 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Athens)

Figure 18 presents the heating and cooling sensible load (needs) patterns of the interacting zones. The multi-zone results show consistent seasonal behaviour across all zones, with heating demands peaking in winter and cooling demands dominating in summer. Thermal Zones 1 and 2, with the same window orientation, exhibit similar energy patterns (aligning with the internal temperature patterns, Figure 15), while zone 3 follows a different cooling demand pattern. The latter is associated with the window orientation (north), which does not contribute the same amount of energy as the south-facing windows in the other zones. This behaviour (also observed in the internal temperature profile above) as expected is explained by the sun's position relative to the building, which is located in the Northern Hemisphere. In addition, since Zone 1 has two neighbouring zones, it is observed to have lower heating and higher cooling energy demands, whereas three out of the four

walls of Zones 2 and 3 are exposed to the external environment. Lastly, similar deviations with the EnergyPlus software to the single-zone case are observed which are also primarily attributed to the inherently different nature of the tools as well as the lack of shading calculations in the OPENBEP4EU Calculation Engine.

To examine the openBEP4EU engine's effectiveness in diverse climatic characteristics, the exact same multi-zone case study has been tested in different European cities. The selected case-study locations are widely distributed across Europe, spanning cold, temperate and Mediterranean climatic areas. Utilising typical meteorological years from Berlin, Madrid, Milan and Stockholm -without imposing any other changes to the engines input-, sensible loads per zone and month were generated to assess whether climate-specific factors impact the consistency of openBEP4EUengine's results. Figure 19 , Figure 20, Figure 21 and Figure 22 present the results per city.

The comparison across Athens, Milan, Berlin, Stockholm and Madrid shows a strong level of agreement for both heating and cooling loads at the monthly level in all three zones. Seasonal trends, peak magnitudes and relative differences between zones are consistent. Small systematic deviations are observed mainly during the summer period, possibly attributed to the influence of thermal mass representation and solar gain distribution differences between the two approaches. Overall, the openBEP4EU engine results demonstrate credible and reliable estimations when benchmarked against EnergyPlus irrespective of climate conditions.

Figure 19 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Madrid)

Figure 20 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Berlin)

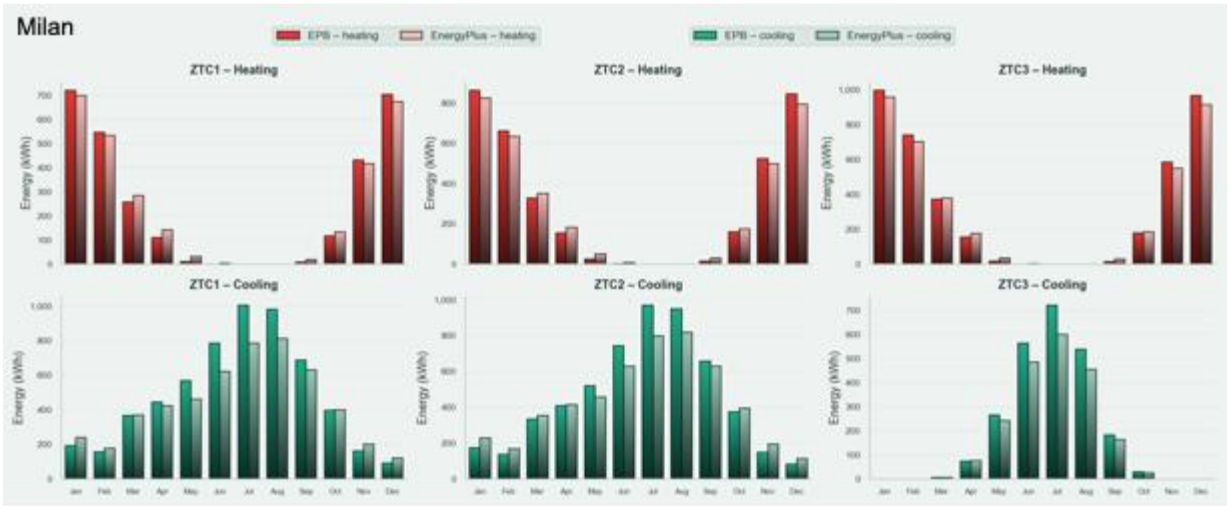


Figure 21 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Milan)

Figure 22 – Multi-zone test case, Heating and Cooling needs: Comparison with EnergyPlus (Stockholm)

4.4 Single-zone air conditioning system

After implementing the calculation of cooling and heating sensible loads, different systems -addressing those demands- have been tested with increasing complexity. Initially, using the results obtained by Case 1, an air-conditioned heating/cooling split unit was placed in the single zone to test an end-to-end scenario of a simulated thermal zone equipped with a technical building system.

Figure 23 – Single-zone implementation top-down view with air conditioning system

The analysis focused on how the system responds -in terms of emission and generation- to the calculated thermal energy needs of the single-zone model. To achieve this, specific performance characteristics (i.e., thermal capacities, coefficients of performance, evaporating and condensing temperature and more) were utilised towards delivering a realistic test case. Indicative HVAC properties are presented in Table 3.

Table 3 – Heat pump properties

Parameter	Symbol	Variable in EPB engine	Unit	Value
Thermal capacity at measurement point for heating $\Phi_{X\,°C}$	$\Phi_{A\,=\,-7\,°C}$	phi_A	kW	1.94
	$\Phi_{B\,=\,2\,°C}$	phi_B	kW	1.29
	$\Phi_{C\,=\,7\,°C}$	phi_C	kW	0.98
	$\Phi_{D\,=\,12°C}$	phi_D	kW	1.17
	$\Phi_{E\,=\,TOL}$	phi_E	kW	1.61
Coefficient of Performance $COP_{X\,°C}$	$COP_{A\,=\,-7\,°C}$	COP_A	-	2.63
	$COP_{B\,=\,2\,°C}$	COP_B	-	4.14
	$COP_{C\,=\,7\,°C}$	COP_C	-	5.17
	$COP_{D\,=\,12\,°C}$	COP_D	-	6.34
	$COP_{E\,=\,TOL}$	COP_E	-	2.09
Thermal Capacity Measurement Point for cooling $\Phi_{X\,°C}$	Φ_{A}=35 °C	phi_A	kW	3.4
	Φ_{B}=30 °C	phi_B	kW	2.51
	Φ_{C}=25 °C	phi_C	kW	1.61
	Φ_{D}=20 °C	phi_D	kW	1.1
Energy Efficiency Ratio $EER_{X\,°C}$	$EER_{A\,=\,35\,°C}$	EER_A	-	3.8
	$EER_{B\,=\,30\,°C}$	EER_B	-	5.95
	$EER_{C\,=\,25\,°C}$	EER_C	-	9.51
	$EER_{D\,=\,20\,°C}$	EER_D	-	15.57
Degradation coefficient	C_d	CD	-	0.25

According to the outcomes of the simulation, the identified pattern for the electric energy demanded by the AC generator were consistent with the input energy needs and HVAC properties. Particularly, the consumed electric energy by the thermal zone corresponded to a portion of the initial needs, dictated by the coefficients of performance (CoP) for heating and energy efficiency ratios (EER) for cooling. As observed in Figure 24 the heating electricity appears on the order of 60–80 kWh/month from the chart. Meanwhile, cooling electricity peaks in summer, roughly around 200–300 kWh/month in the hottest months. Both outcomes are plausible for a 48 m² zone and located in Athens.

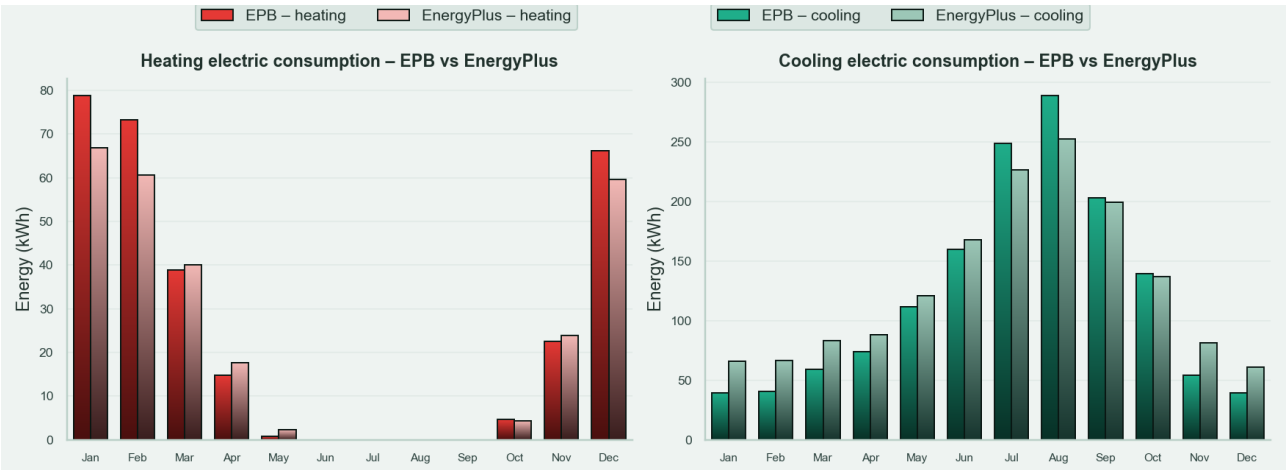


Figure 24 – Single-zone test case with air conditioning system: electric energy consumed by the system for heating and cooling

4.5 Multi-zone air-conditioning system

Going a step further, by incorporating the AC split single-zone simulation of Section 4.4 into the multizone sensible loads simulation of section 4.3, a combined effect on the total required electric energy was examined (Figure 25). In detail, the heating and cooling needs provided for three zones were addressed by three separate AC split units installed in each zone, i.e., one indoor and one outdoor unit per zone. As a result, three distinct heating emitters where matched to three distinct generators and similarly three cooling emitters were associated with three separate generators. The outcomes are presented in Figure 26.

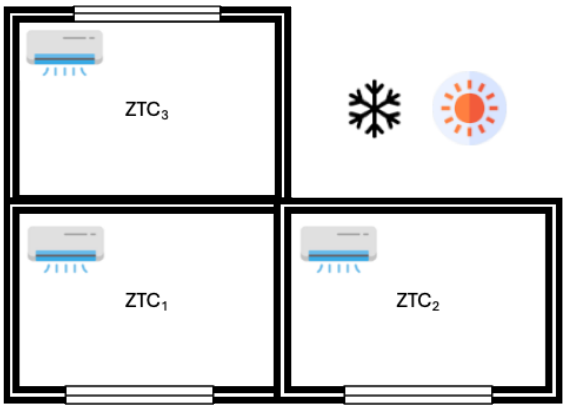


Figure 25 – Multizone implementation top-down view with A/C splits

The generated results exhibited consistency with both the single-zone AC split unit and multizone sensible loads profiles.

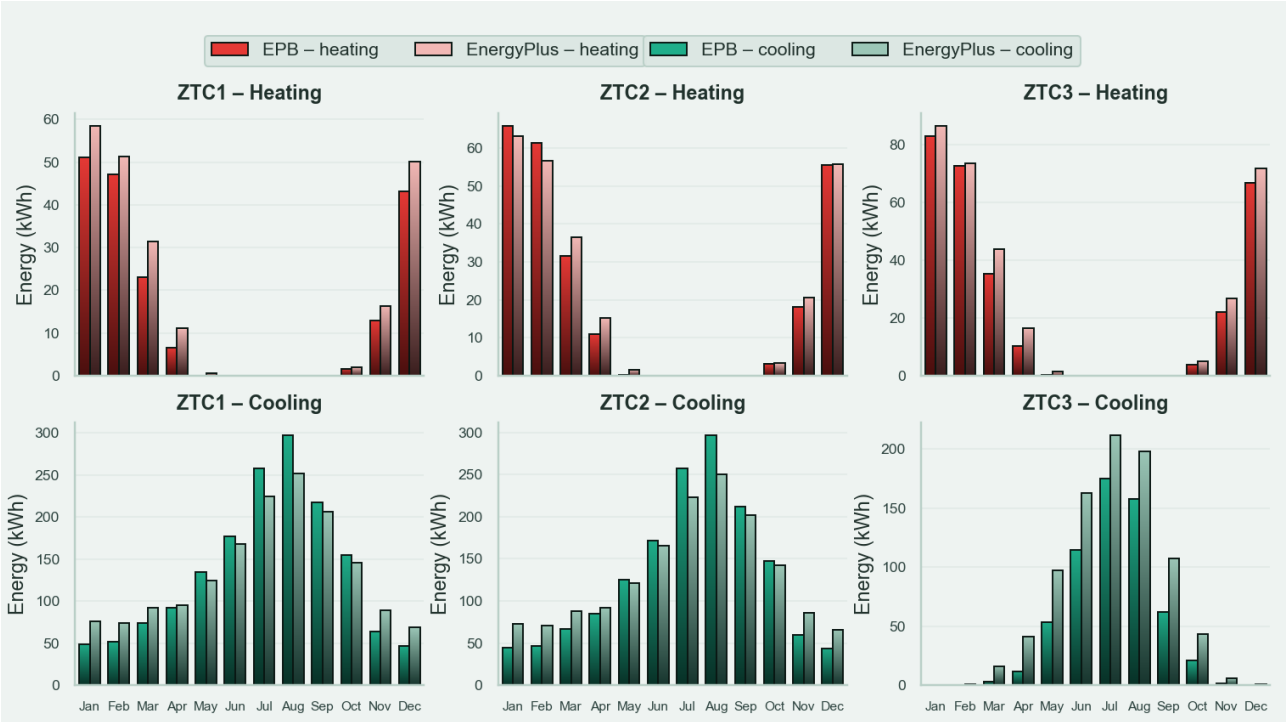


Figure 26 – Multi-zone test case with air conditioning system: electric energy consumed by the systems for heating and cooling

4.6 Multi-zone heating and domestic hot water with gas boiler

A more advanced scenario was formulated to add complexity to the openBEP4EUCalculation Engine and illustrate the interactions between different systems and zones. Building on the three-zone configuration, both the energy and domestic hot water needs were calculated to act as a basis for a realistic end-to-end simulation of a building with multiple zones and technical building systems. In Figure 27, the pie chart summarizes the share of total annual energy demand per heating, cooling and Domestic Hot Water needs.

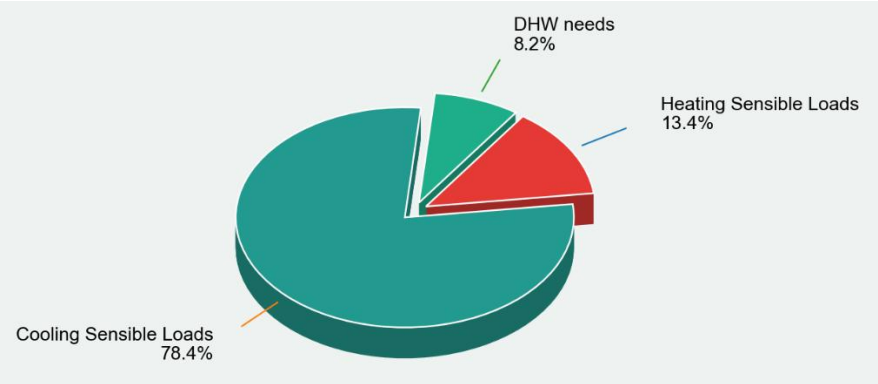


Figure 27 – Annual share by type of energy needs

Figure 27 highlights the dominance of the cooling sensible loads mainly due to the climatic characteristics of Athens as well as the orientation of the installed windows in the three zones.

In this extended simulation, to address the heating needs, each zone was equipped with a radiator supplied with hot water from a common distribution network circulating all zones. The hot water entering the common piping system was heated by a gas boiler. Cooling loads were not addressed under this case.

For the Domestic Hot Water system accounting for the hot water needs, a separate hot water distribution network was selected for each zone i.e., three different pipes with the same length and attributes sharing the same storage system. Finally, the DHW storage tank was connected to the same gas boiler that heats the water for the heating system. An indicative top-down design is depicted in Figure 28 which visualizes the configuration of the multizone building, as well as the heating and domestic hot water systems that are modelled in case 6. The colour coding of the piping system in Figure 28 represents the distribution of water temperatures across the different sections of the piping network.

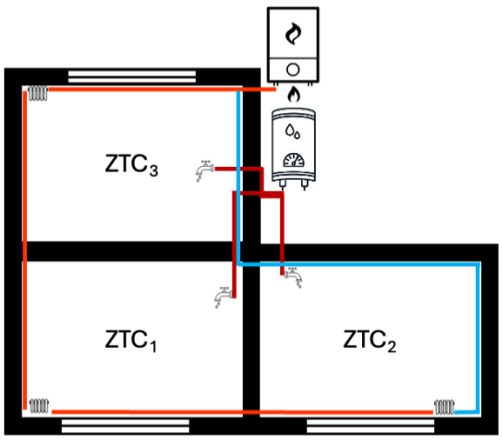


Figure 28 – Top-down view of the multi-zone heating and domestic hot water with gas boiler test case

Due to the complexity of the simulation comprising multiple subsystems, a multitude of input variables had to be defined concerning emission, distribution, storage and generation. Table 4 includes the required parameters along with the respective standard symbols, values and underlying subsystem type. The input maintains a standard-compliant structure meaning the parameters directly align with EPB standards-based methodologies, ensuring traceability and consistency with ANNEX A.1 – Input Variables where the entirety of input/output parameters originating from the 52000 family and associated standards is presented.

Table 4 – Multi-zone heating and domestic hot water with gas boiler test case: Input parameters

Parameter	Symbol	Variable in EPB engine	Unit	Value
temperature variation based on not balanced hydraulic systems	Δθhydr	delta_theta_hydr	K	0.6
control variation of temperature	Δθctr	delta_theta_ctr	K	0.7
temperature variation based on intermittent controls operation system	Δθim,crt	delta_theta_im_ctr	K	-0.3
temperature variation based on intermittent operation of the emission system	Δθim,emt	delta_theta_im_emt	K	-0.3
temperature variation based on radiation by type of the emission system	Δθrad	delta_theta_rad	K	0
temperature variation based on the stratification	Δθstr	delta_theta_str	K	1.2
temperature variation based on an additional heating/cooling loss by emitters embedded in the envelope	Δθemb	delta_theta_emb	K	0
temperature variation for consideration of solar and internal gains	Δθe,sol	delta_theta_e_sol	K	0
temperature variation based on room automation	Δθroomaut	delta_theta_roomaut	K	-0.5
Radiant factor	RF	RF	-	0
Room height	h_a	ha	m	2.7
DHW distribution temperature (hot water supply)	$\theta_{W;dis}$	theta_W_dis	°C	60
Cold water inlet temperature	$\theta_{W;cold}$	theta_W_cold	°C	5
Indoor temperature of water service area	$\theta_{int;wsa,j;m,i}$	theta_int_wsa_j_m_i	°C	20
Design outdoor temperature	$\theta_{H;e;des}$	theta_H_e_des	°C	-16
Outdoor temperature at no-heating condition	$\theta_{H;e;nh}$	theta_H_e_nh	°C	16
Design supply (flow) temperature	$\theta_{H;flw;des;Hsa,j}$	theta_H_flw_des_Hsa_j	°C	70
Supply temperature at no-heating condition	$\theta_{H;flw;nh;Hsa,j}$	theta_H_flw_nh_Hsa_j	°C	35
Design return temperature	$\theta_{H;ret;des;Hsa,j;m,i}$	theta_H_ret_des_Hsa_j	°C	50
Return temperature at no-heating condition	$\theta_{H;ret;nh;Hsa,j;m,i}$	theta_H_ret_nh_Hsa_j	°C	28
External pipe diameter	D_{ext}	D_ext	m	0.022
Insulation thickness	$t_{ins;m}$	t_ins_m	m	0.013
Insulation thermal conductivity	λ_D	lamda_D	W/m·K	0.035
Concrete thermal conductivity	λ_G	lamda_G	W/m·K	1.4
Pipe depth in slab	z	z_m	m	0.05
Volume of storage layers	$V_{;sto;vol}$	V_sto_vol	m3	45 / 45 / 30 / 30
Storage diameter	D_{sto}	D_sto	m	0.5
Storage height	H_{sto}	H_sto	m	0.76
Cold water temperature	$\vartheta_{W;cold}$	theta_W_cold	°C	11.2
Minimum temperature for dhw	$\vartheta_{W;out;min}$	theta_W_out_min	°C	45
heater position 1 means bottom	N_{bu}	Nbu	-	1
Back-up heater ON (°C)	$\vartheta_{sto;set;on;bu}$	theta_sto_set_bu_on	°C	57
Set temperature (°C)	$\vartheta_{sto;set;off;bu}$	theta_sto_set_bu_off	°C	60
Maximum power input from heater	$\Phi_{sto;bu;max}$	fi_sto_in_max	kW	1.5
drop off position for heating (loosing energy/temp)	N_h	Nh	-	3
Minimum temperature for heating	$\vartheta_{H;out;min}$	theta_H_out_min	°C	0
General classification of heat generator	GEN_TYP	generation_type	-	"Standard boiler"
Specific boiler category	GEN_BURN	boiler_type	-	"Atmospheric gas boiler up to 250 kW"
Affects default efficiencies & factors	N/A	build_year	-	"after 1994"

Fuel type used	GEN_FUEL	energy_carrier	-	"Natural gas"
Nominal heat output	P_n	Pn	kW	10
Mean heating circuit temperature	$\vartheta_{Hc;mn}$	theta_HC_mn	°C	48.9
Mean boiler water temperature	$\vartheta_{Wc;mn}$	theta_W_mn	°C	50

The outcomes of the boiler simulation are fundamentally different to the AC split unit results as the energy demanded by the boiler is delivered by a gas-based combustion engine. Consequently, the calculated energy (Q_gen_in) is expected to significantly exceed the initial building energy needs, as energy is progressively accumulated across the successive subsystems of the modelling chain due to system losses and auxiliary operation. Figure 29 showcases this cumulative behaviour. In more detail, the emission demands (Q_em_in) are augmented compared to the sensible loads, as they include radiator losses and auxiliary operation. Subsequently, the distribution subsystem further increases the required energy due to thermal losses along the piping network. Additionally, the storage subsystem increases the amount of energy consumed to address energy losses within the tank. Lastly, the generation subsystem accounts for all upstream demands and associated losses, resulting in the highest energy requirement within the system chain.

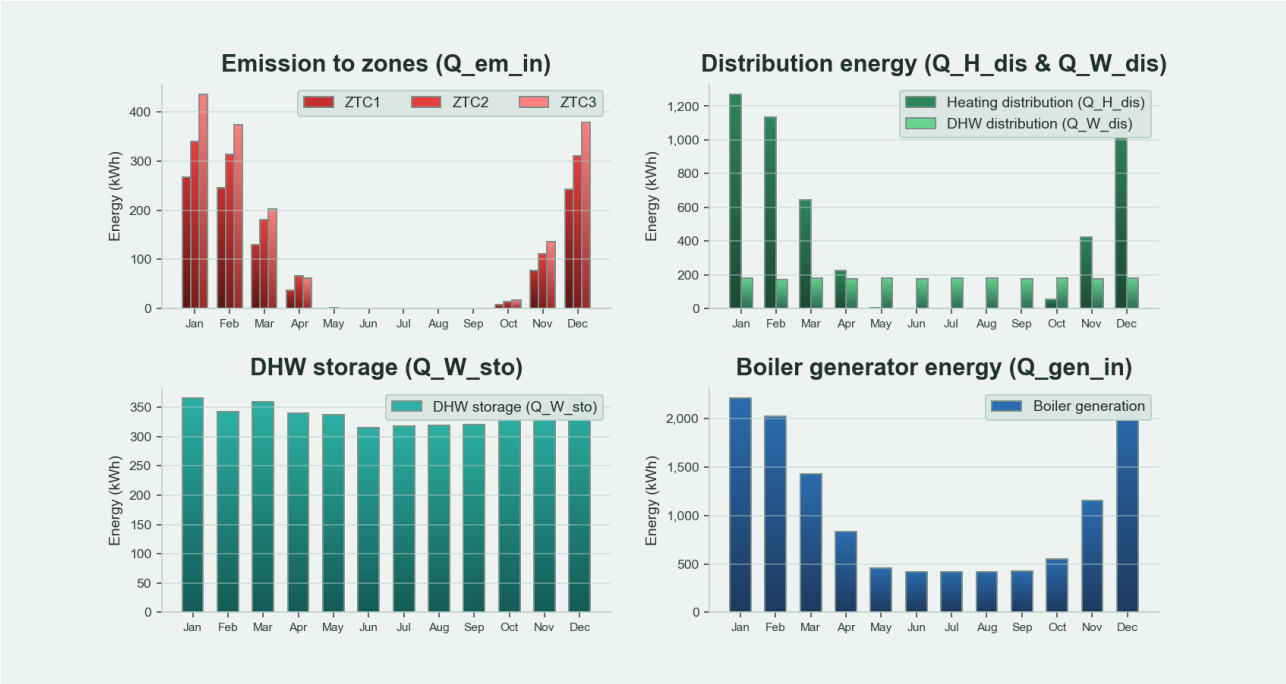


Figure 29 – Multi-zone heating and domestic hot water with gas boiler test case: emission, distribution, storage, and generation results

In contrast to the disaggregated building energy demand across the thermal chain, Figure 30 presents the monthly allocation of total energy consumption using a stacked bar chart. The energy needs (i.e., heating and dhw needs) are stacked together with additional contributions from emission, distribution, storage and generation subsystems. The latter represent auxiliary electricity consumption, system heat losses and any recoverable energy within the technical systems. The sum of energy needs along with the additional contributions from each subsystem constitute the total amount of energy that needs to be delivered by the gas boiler.

Figure 30 includes a stacked bar chart which illustrates, monthly, the extra energy introduced at each stage of the modelling chain (i.e., focusing exclusively on losses and auxiliary consumption), rather than the cumulative energy transferred between subsystems (depicted in Figure 29). It highlights how losses and auxiliary demands evolve throughout the year in response to seasonal conditions

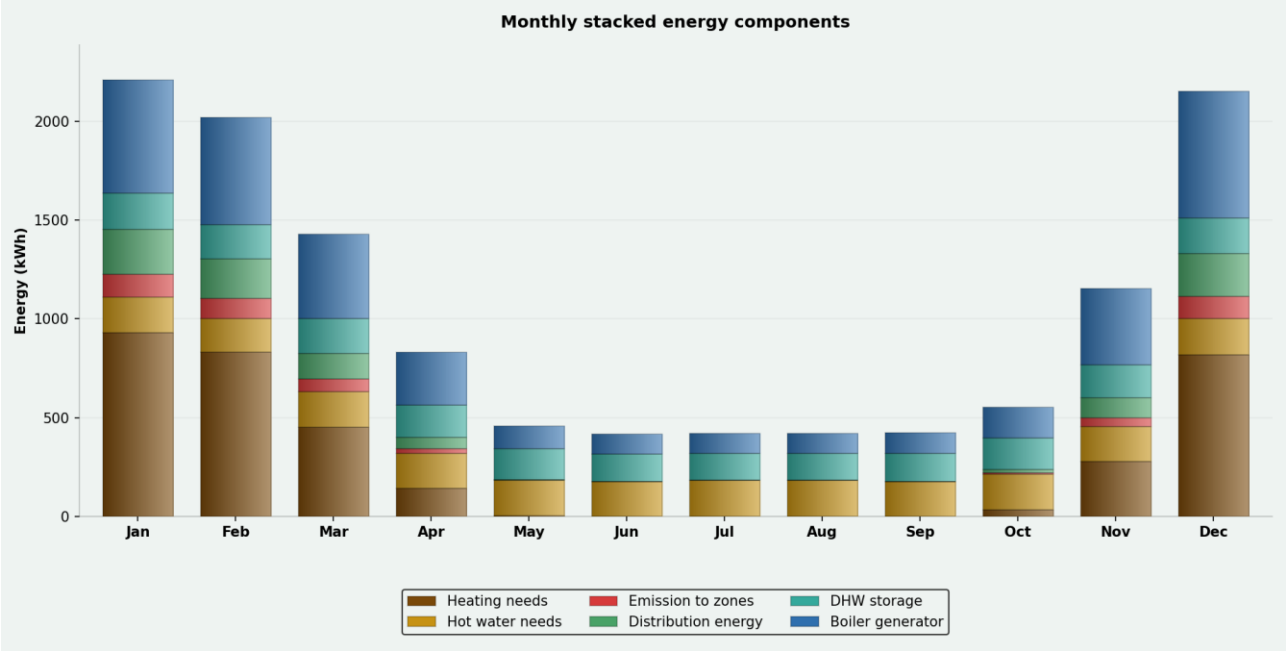


Figure 30 – Multi-zone heating and domestic hot water with gas boiler test case: Monthly allocation of total energy consumption across thermal zone demands and subsystem contributions

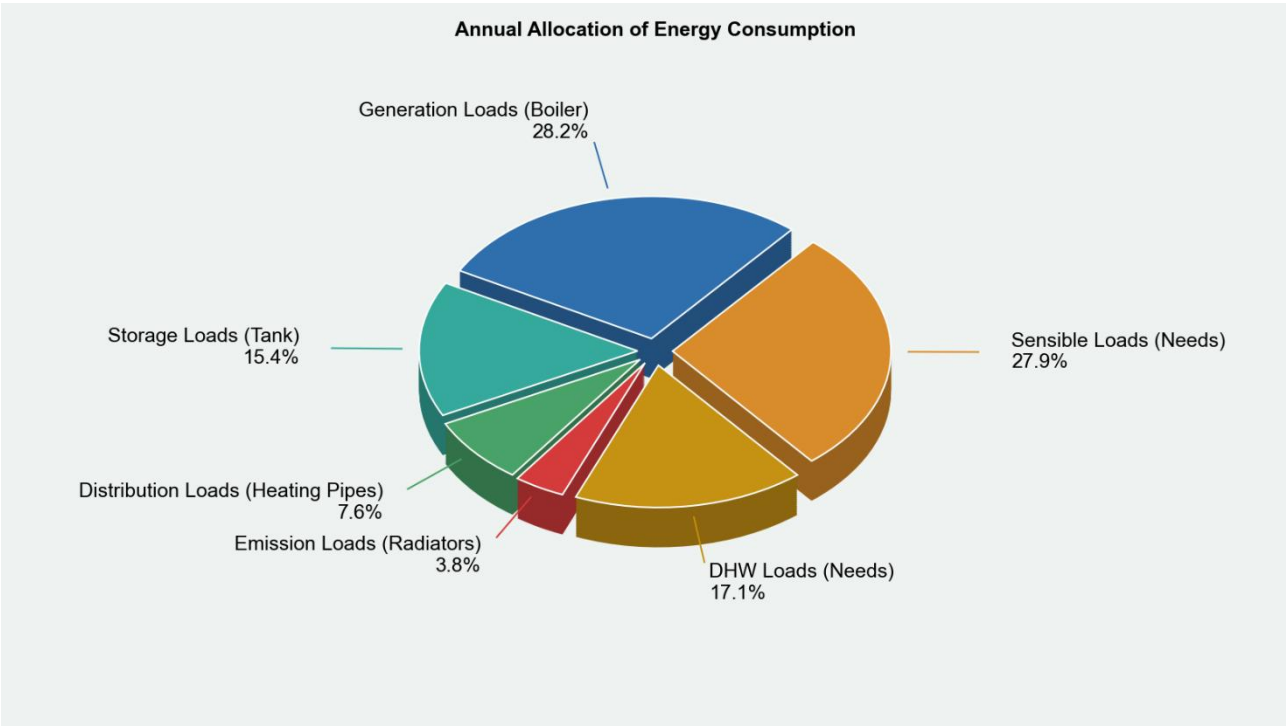


Figure 31 – Multi-zone heating and domestic hot water with gas boiler test case: Annual allocation of total energy consumption across thermal zone demands and subsystem contributions

In contrast, Figure 31 provides an annual aggregation of these components, facilitating a clear comparison of their relative contribution to the total energy consumption. According to Figure 31, the least dominant additional contributions correspond to the emission and distribution subsystems. This is justified by the fact that, for the

distribution of hot water, embedded insulated pipes within the building have been considered, thereby minimizing heat losses. Similarly, the emission subsystem includes only one radiator per thermal zone, also resulting in limited thermal losses. In contrast, the storage subsystem exhibits a more significant contribution due to thermal inertia effects and the frequent operation of the domestic hot water (DHW) system (approximately 3 hours per day), which increases energy consumption to maintain the desired temperature levels. Lastly, the generation subsystem presents the highest additional energy demand, as the selected gas boiler (combustion engine) operates with relatively low efficiency (<30%), leading to increased fuel consumption to compensate for upstream losses and system inefficiencies.

5 CONCLUSIONS

Deliverable D2.2 documents the development of the OpenBEP4EU Calculation Engine, developed by QUE within WP2, Task 2.2 of the OpenBEP4EU project. The work addresses a clear need emerging from the EPBD recast, which requires building energy performance to be calculated through methodologies aligned with the EPB standards framework while, at the same time, national implementation remains fragmented in terms of standards coverage, calculation procedures, software environments, and interpretation choices. In this context, the OpenBEP4EU Calculation Engine has been developed as a transparent and reusable calculation core intended to support a more consistent software implementation of the EPB calculation chain.

A central contribution of the work has been the structured transition from standards to software. This included the reconstruction of the conceptual flow of the EPB standards, the assessment of their programmability, and the detailed analysis of equations, variables, and input dependencies needed to translate normative content into executable logic. This process provided the methodological basis for implementation prioritisation and for the architecture of the resulting engine. It also made it possible to distinguish clearly between standards that lend themselves to direct software implementation and standards that primarily support interpretation, documentation, inspection, or quality assurance.

On that basis, the current version of the engine establishes the functional core of the EPB calculation workflow within the present project scope. The implemented framework already supports the modelling of key parts of the building thermal balance, including hourly thermal energy needs, thermal properties of envelope elements, selected ground-contact cases, climate-related inputs, and operational profiles, together with technical building systems for heating, cooling, and domestic hot water. The resulting architecture is modular and layered, separating data import, validation, transformation, calculation modules, and orchestration. This design is important both from a software-engineering perspective and from the perspective of future reuse and extension.

The deliverable has also shown that the calculation engine is not treated as an isolated research prototype, but as the basis of a reusable software asset. In particular, the core calculation modules are intended to be made publicly available through the project GitHub repository as a library of reusable Python components, accompanied by the corresponding dependency specification. In this sense, the work reported in D2.2 does not only document an internal implementation effort, but establishes the basis for a public calculation core that can be inspected, reused, and further extended beyond the project context.

The preliminary testing presented in this deliverable has provided an important first assessment of the implemented engine. The approach combined reference-based comparison against EPB spreadsheet material with comparative assessment against EnergyPlus software across a set of progressively more demanding test cases. These tests do not claim formal validation of the full framework; however, they show that the OpenBEP4EU Calculation Engine produces numerically consistent and technically plausible results, and that its outputs follow the expected magnitude and temporal behaviour under different modelling configurations. This provides confidence that the implemented core is sufficiently robust to support the next development stage.

At the same time, D2.2 makes transparent the current implementation boundaries of the engine. Some standards and system domains remain only partially addressed, while others are planned for subsequent releases. In particular, further development is needed to extend system coverage, complete the remaining technical modules, and broaden the range of cases covered by the framework. These next steps are already foreseen in the project roadmap and will build directly on the modular basis established in the present deliverable.

Overall, D2.2 marks an important milestone in the OpenBEP4EU project. It delivers the first implementation of the calculation core, documents the methodological route followed from EPB standards to software, and demonstrates through preliminary test cases that the approach is operational. The work completed so far

provides a solid foundation for the upcoming releases, in which the engine will be expanded, consolidated, and further aligned with the broader OpenBEP4EU platform.

6 References

[1] European Parliament and Council of the European Union, “Directive (EU) 2024/1275 on the energy performance of buildings (recast),” Official Journal of the European Union, 2024.

[2] European Commission, “Commission notice – Annex 12: Common general framework for the calculation of the energy performance of buildings (Annex I),” Publications Office of the European Uni, 2025a.

[3] European Commission, “Commission notice – Annex 3: Energy performance certificates (Articles 19–21 and Annex V),” Publications Office of the European Union, 2025b.

[4] Concerted Action EPBD, “Outcomes and reports,” 2026. [Online]. Available: https://www.ca-epbd.eu/outcomes .

[5] Concerted Action EPBD, “Certification and training – Status in 2022 (Report CT5),” CA EPBD , 2022.

7 ANNEX A

This annex presents the parameters identified during the analysis and implementation of the ISO 52000 family -and associated- standards within the openBEP4EU Calculation Engine. The table provides a structured overview of the input and output variables defined in the standards, including their corresponding descriptions, standard and engine symbols and units. It further indicates whether each parameter is matched with an informative value ("IV" column) according to Annex B of the standards. Finally, column "oCE" specifies whether each parameter has been incorporated into the openBEP4EU engine.

7.1 ANNEX A.1 – Input Variables

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
52031-1	control variation of temperature	$\Delta_{\theta ctr}$	delta_theta_ctr	°C	√	√
	temperature variation based on control, not certified products	$\Delta_{\theta ctr,1}$	delta_theta_ctr_1	°C	√	√
	temperature variation based on control, certified products	$\Delta_{\theta ctr,2}$	delta_theta_ctr_2	°C	√	√
	hysteresis of thermostatic valve	θ_H		–	√	
	effect of supply water temperature on thermostatic controllers (TRV) head sensing element	θ_W		–	√	
	temperature variation based on not balanced hydraulic systems	$\Delta_{\theta hydr}$	delta_theta_hydr	°C	√	√
	temperature variation based on intermittent controls operation system	$\Delta_{\theta im,crt}$	delta_theta_im_ctr	°C	√	√
	temperature variation based on intermittent operation of the emission system	$\Delta_{\theta im,emt}$	delta_theta_im_emt	°C	√	√
	temperature variation based on radiation by type of the emission system	$\Delta_{\theta rad}$	delta_theta_rad	°C	√	√
	temperature variation based on the stratification	$\Delta_{\theta str}$	delta_theta_str	°C	√	√
	temperature variation based on the stratification — part of influence due to "over-temperature"	$\Delta_{\theta str,1}$	delta_theta_str_1	°C	√	√
	temperature variation based on the stratification — part of influence due to "specific heat losses via external components"	$\Delta_{\theta str,2}$	delta_theta_str_2	°C	√	√
	temperature variation based on an additional heating/cooling loss by emitters embedded in the envelope	$\Delta_{\theta emb}$	delta_theta_emb	°C	√	√
	temperature variation based on an additional heating/cooling loss by emitters embedded in the envelope — part of influence due to the "system"	$\Delta_{\theta emb,1}$	delta_theta_emb_1	°C	√	√
	temperature variation for consideration of solar and internal gains	$\Delta_{\theta e,sol}$	delta_theta_e_sol	°C	√	√
	temperature variation based on an additional heating/cooling loss by emitters embedded in the envelope — part of influence due to "specific heat losses via laying surfaces"	$\Delta_{\theta emb,2}$	delta_theta_emb_2	°C	√	√
	temperature variation based on room automation	$\Delta_{\theta roomaut}$	delta_theta_roomaut	°C	√	√
	radiant factor of radiant heaters for room heights ≥4 m	RF	RF	–	√	√
	room height	h_R	room_height	m	√	√
	electrical rated power consumption of the control	P_{ctr}		–	√	
	electrical rated power consumption of the equipment	$P_{H,aux}$	p_h_aux	W	√	√
	electrical rated power consumption of the fan	P_{fan}	p_fan	W	√	√
	design nominal useful emitter power	Φ_{Hemn}		–	√	

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
[illegible]	Thermal capacity at mesurement point X °C	Φ_x	phi_X	kW	√	√

	COP at mesurement point X	COP_x	COP_X	–	√	√
	Degradation coefficient	C_d	cd	–	√	√
	Test point room air temperature	$q_{in,X}$	theta_in_X	°C	√	√
	Test point outdoor air temperature	$q_{e,X}$	theta_e_X	°C	√	√
	Temperature difference between evaporation and indoor air at capacity A (cooling)	$Dq_{evap;ref}$	delta_theta_evap_ref	°C	√	√
	Maximum evaporation temperature	$q_{evap;max}$	theta_evap_max	°C	√	√
	Temperature difference between condensation and outdoor air temperature at capacity A (cooling)	$Dq_{cond;ref}$	delta_theta_cond_ref	°C	√	√
	Minimum condensing temperature	$q_{cond;min}$	theta_cond_min	°C	√	√
	Temperature difference between evaporation and external air at reference capacity (heating)	$Dq_{evap;E}$	delta_theta_evap_E	°C	√	√
	Temperature difference between condensation and room air temperature at reference capacity (heating)	$Dq_{cond;E}$	delta_theta_cond_E	°C	√	√
	Coefficient for calculation of minimum/maximum capacity	A_{cap}	A_cap	–	√	√
	Coefficient for calculation of minimum/maximum capacity	B_{cap}	B_cap	–	√	√
	Coefficient for the calculation of RCE efficiency	A_{RCE}	A_RCE	–	√	√
	Coefficient for the calculation of refrigerant cycle efficiency	B_{RCE}	B_RCE	–	√	√
	EER at measurement point X	EERx	EER_X	–	√	√

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
15316-4-1	Fuel type	GEN_FUEL	energy_carrier	List	√	√
	Generator Type	GEN_TYP	generator_type	List	√	√
	Burner type	GEN_BURN	boiler_type	List	√	√
	Boiler location	TH_ZONE		List		
	Type of control	HEAT_GEN_CTRL		List		
	Generation circuit typology	GEN_CIRC_TY POL		List		
	Number of dwellings within a building	Nflat		–		
	Number of peak tappings per day	nSp		–		
	Heat load	Φh		kW		
	Rated output for cooling system	Pn,C		kW		
	Rated output for ventilation system	Pn,V		kW		
	Rated output for DHW system	Pn,W		kW		
	Heat input to heating distribution system	QH,dis;in	Q_H_gen_out	kWh		√
	Heat input to cooling distribution system	QC,dis;in		kWh		
	Heat input to ventilation distribution system	QV,dis;in		kWh		
	Daily energy need for domestic hot water	QW,b,d	DHW_needs	kWh		√
	Heat input to DHW distribution system	QW,dis;in	Q_W_gen_out	kWh		√
	Usage period for heating	tH;use		h or d		
	Usage period for cooling	tC;use		h or d		
	Usage period for ventilation	tV;use		h or d		
	Usage period for domestic hot water	tW;use		h or d		
	Running time for heating	tH	t_H	s/h or h/mth		√
	Running time for cooling	tC		s/h or h/mth		
	Running time for ventilation	tV		s/h or h/mth		
	Running time for DHW	tW	t_w	s/h or h/mth		√
	External air temperature	ϑe	theta_ext	°C		√
	Daily average design temperature	ϑe;min		°C		
	Generator average water temperature	ϑHc;mn	theta_HC_mn	°C		√
	Average return temperature to generator	ϑHc;RT		°C		
	Generator avg water temperature (cooling)	ϑCc;mn		°C		
	Generator avg water temperature (ventilation)	ϑVc;mn		°C		
	Generator avg water temperature (DHW)	ϑWc;mn	theta_W_mn	°C		√
	Average return temperature (cooling)	ϑCc;RT		°C		
	Average return temperature (ventilation)	ϑVc;RT		°C		
	Average return temperature (DHW)	ϑWc;RT		°C		

	Average return temperature to generator	ϑgen;RT		°C		
	Ambient temperature	ϑbrm	theta_brm	°C	√	√
	Cold water temperature	ϑk		°C		
	Delivered energy input of generation subsystem	Egen;del;in		kg, m³		
	Generator output at full load	Pn	Pn	kW		√
	Generator output at intermediate load	Pint	b_pint	kW	√	√
	Temperature difference (return vs flue gas)	Δϑwfg		°C		
	Full load efficiency	ηPn;add	BoilerParams	–	√	√
	Mean water temperature at full load	ϑgen;test;Pn;add	theta_gen_test_Pn	°C	√	√
	Part load efficiency	ηPint;add	BoilerParams.pint.n	–	√	√
	Mean water temperature at part load	ϑgen;test;Pint;add	theta_gen_test_Pint	°C	√	√
	full load correction factor	fcorr;Pn	f_corr_pn	–	√	√
	part load correction factor	fcorr;Pint	f_corr_pint	–	√	√
	control factor	fctr;ls	f_ctr_ls	–	√	√
	auxiliary energy transmitted to the distribution sub-system	faux;rvd	f_aux_rvd	–	√	√
	domestic hot water type	dhw_type	dhw_type	–	√	√
	coefficients	c1,…c8	BoilerParams.c1 … c8	–	√	√
	Ratio of gross calorific value to net calorific value conversion factor	fHs/Hi	f_Hs_Hi	–	√	√
	values for Instantaneous electrical water heater	*ftr;ls*	f_tr_ls	–	√	√
	Temperature reduction factor	fbrm	f_brm		√	√

Stand-ard	**Variable Description**	**Symbol**	**Variable name in EPB Engine**	**Unit**	**IV**	**oCE**
52032-1	Pipe outer diameter	Dext	Dext_m	m		√
	insulation conductivity	λD	lambda_D	W/mK		√
	external heat transfer coefficient	ha	h_a	W/m²·K		√
	temperature difference driving term	ΔT	DT	K		√
	Wall conductivity	λG	lambda_G	W/mK		√
	Depth within wall	z	z_m	m		√
	Distance between pipe axis	e	e	m		√
	Required output of heating emission	Q;em;in	Q_em_in	kW		√
	Outdoor DESIGN temperature	θH;e;des	theta_H_e_des	°C		√
	Outdoor NO-HEATING (balance) temperature	θH;e;nh	theta_H_e_nh	°C		√
	Flow design temp at generator	θH;flw;des;Hsa,j	theta_H_flw_des_Hsa_j	°C		√
	Flow temp at no-heating	θH;flw;nh;Hsa,j	theta_H_flw_nh_Hsa_j	°C		√
	Return design temp	θH;ret;des;Hsa,j;m,i	theta_H_ret_des_Hsa_j	°C		√
	Return temp at no-heating	θH;ret;nh;Hsa,j;m,i	theta_H_ret_nh_Hsa_j	°C		√
	Total operation time of heating	tH;dis;Hsa,j,m,i	t_H_dis_Hsa_j_m_i	h		√
	auxiliary energy fraction	kH;dis;aux;Hsa,j	k_H_dis_aux_Hsa_j			√
	Recovered fraction of recoverable losses	kH;ls;rvd;Hsa,j	k_H_ls_rvd_Hsa_j			√
	fraction of auxiliary recovered	kH;dis;aux;rvd;Hsa,j	k_h_dis_aux_rvd_Hsa_j			√
	external temperature per thermal zone at time	θe;H;ztc,k;m,i	theta_e_H_ztc_k_m_i	°C		√
	insulation thickness around pipe	s	t_ins_m	m		√
	Pipe length	L	l_p_k	m		√
	Surrounding temperature of pipes	θsur;p,k;m,i	theta_sur	°C		√
	Power of circulation pump	PW;dloop;aux	P_W_dloop_aux	kWh		√
	Recovery factor of auxiliary energy	kW;dis;aux;rh	k_W_dis_aux_rh			√
	DHW distribution loop operating time fraction	kW;op;Wsa,j	k_W_op_Wsa_j			√
	Recoverable fraction of distribution losses	kls;rbl;p,k	k_ls_rbl_p_k			√
	Temperature of distribution loop	θW;dis	theta_W_dis	°C		√
	Cold water temperature	θW;cold	theta_W_cold	°C		√
	the water service area indoor temperature	θint;wsa,j;m,i	theta_int_wsa_j_m_i	°C		√

	Total operation time of dhw loop	tW;dloop;Wsa,j	t_W_dloop_Wsa_j	h		√
	DHW preparation period	tW;m,i;dd	t_W_m_i_dd	h		√
	Number of tappings per day	ntap;day;Wsa,j	ntap_day_Wsa_j			√
	Volume of water in the distribution pipe	VW;p,k	V_W_p_k	m3		√
	Required output of dhw final distribution	QW;dfin;out;Wsa,j	Q_W_dfin_out_Wsa_j	kW		√
	Temperature reduction factor	bsur	b_sur			√
	Specific heat of water	cw	cw	kWh/(kg·K)	√	√
	Density of water	ρw	Rw	kg/m3	√	√

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
12831-3	Effective size of the hot water storage tank	Vsto		l		
	Height of the temperature sensor in the storage tank	hsensor		m		
	Loading factor	fl		–		
	Standby heat loss of the hot water tank	qsb,sto		kWh/d		
	Time lag of the heat generation system (from the perspective of water heating)	tlag		min	√	
	Floor area	A	building_area	m²		√
	Maximal water temperature in the storage tank used for design purposes	ϑw,sto,max		°C		
	Type of building (building category)	Building type		List	√	√
	Density of water	ρw		kg/m3	√	√
	Specific thermal capacity of water	cw		kJ/kgK	√	√
	Delivery water temperature	ϑW,draw	teta_W_draw	°C		√
	Cold water temperature	ϑW,c	teta_W_cold	°C	√	√
	Number of units (employees)	nu	number_of_person		√	√
	Reference hot water temperature	θW;h;ref	teta_w_h_ref	°C	√	√
	Reference cold water temperature	θW;c;ref	teta_w_c_ref	°C	√	√
	Specific energy need	qW,b,d	area_specific	kWh/m² d	√	√
	Reference hot water temperature	θW;h;ref	teta_w_h_ref	°C	√	√
	Specific need	VW;f;day	V_W_f_day	l/u day		√
	Number of workdays in the year	nday;wrk		-		√
	Number of Saturdays in the year	nday;sat		-		√
	Number of holidays in the year	nday;hol		-		√
	Total number of days of the year	nday;tot		-		√
	Relative fraction of daily needs.	xW;nd;d;h,i	x_q_h_i_coor		√	√

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
15316-5 EPB spreadsheet	Volume of storage layers	V;sto;vol	V_sto_vol	lt		√
	initial temperatures of layers	ϑsto;t-1;vol,i	theta_sto_ini_vol	°C	√	√
	Storage diameter	Dsto	D_sto	m		√
	Storage height	Hsto	H_sto	m	√	√
	Stand-by losses coefficient	Hsto;ls	H_sto_ls	W/K		√
	Connection losses factor	Hsto;conn;ls,i	H_sto_conn_ls_layer	W/K		√
	water density	rw	water_density	kg/m^3	√	√
	Cold water temperature	ϑW;cold	theta_W_cold	°C	√	√
	water specific heat	Cp;w	C_w	Wh/(kg·K)	√	√
	Minimum temperature for dhw	ϑW;out;min	theta_W_out_min	°C		√
	Distribution loop losses	QW;dis;ls;loop	Q_W_dis_ls_loop	kWh		√
	Return layer of circulation loop	Nloop	N_loop			√
	Circulation loop active	W_LOOP_ON	Circulation_loop_active			√
	heater position 1 means bottom	Nbu	Nbu			√
	Back-up heater ON	ϑsto;set;on;bu	theta_sto_set_bu_on	°C		√

	Set temperature	ϑsto;set;off;bu	theta_sto_set_off_bu	°C		√
	Maximum power input from heater	Φsto;bu;max	fi_sto_in_max	kW	√	√
	location of solar heater	Nsol	Nsol			√
	energy input from solar	Qsto;sol;in	Q_sto_sol_in	kWh	√	√
	required energy for heating	QH;sto;out;req	Q_H_sto_out_req	kWh	√	√
	drop off position for heating (loosing energy/temp)	Nh	Nh			√
	Minimum temperature for heating (°C)	ϑH;out;min	theta_H_out_min	°C		√
	Nominal mass flow rate back-up heater	ṁbu;n	m_bu	kg/s	√	√
	Mass flow rate solar loop	ṁsol	m_sol	kg/s	√	√
	Heat transfer rate of the solar colector heater heat exchanger	Hexh;SC	H_exh_sol	W/K	√	√
	Heat transfer rate of the back-up heater heat exchanger	Hexh;BH	H_exh_bu	W/K	√	√
	Calculation interval	tci	t_ci	h		√
	installation room coefficient	bsto	b_sto			√
	installation room overtemperature	Dqbrm	delta_theta_brm	°C		√
	Storage losses correction factor	fsto;ls	f_sto_ls		√	√
	Energy required for DHW	QW;sto;out;req	Q_sto_out_req	kWh		√
	internal temperature of the zone	ϑint	theta_int	°C		√
	external temperature of the zone	θext	theta_ext	°C		√
	Number of loops	Nloop	N_loop			√

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
13370	Area of floor	A	area	m²		√
	Exposed perimeter	P	exposed_perimeter	m		√
	Length of linear thermal bridge	lk		m		
	Thickness of the walls of the dwelling	dw,e	wall_thickness_for_ground	m		√
	Height of the upper surface of the floor above external ground level	h	room_height	m		√
	Depth of basement floor below ground level	z		m		
	Indoor environment temperature in the building under consideration	θint	Theta_int	°C		√
	External air temperature	θe	Theta_ext	°C		√
	Virtual temperature of the ground	θvi,m	Theta_gr_ve	°C		√
	Thermal transmittance of element *i*	Ui	thermal_transmittance	W/(m²·K)		√
	Linear thermal transmittance associated with wall/floor junction	Ψk	psi_k	W/(m·K)		√
	Specific heat capacity of air at constant pressure	cp	cp_air	J/(kg·K); Wh/(kg·K)	√	√
	Density of air (at 10 °C and 100 kPa pressure)	ρ	rho_air	kg/m³	√	√

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
6946:2017	area	A		m2		√
	thickness of material layer	d	thickness_per_layer_matrix	m		√
	design thermal conductivity	λ	design_thermal_conductivity_matrix	W/(m·K)		√
	external surface resistance	Rse	external_surface_resistance	m2·K/W		√
	internal surface resistance	Rsi	internal_surface_resistance	m2·K/W		√
	thermal resistance of unheated spaces	Ru		m2·K/W		X
	thermal resistance of air layer	Ra		m2·K/W		X
	thermal resistance of unventilated air layer	Rtot;u		m2·K/W		X
	thermal resistance of ventilated air layer	Rtot;c		m2·K/W		X
	radiative coefficient for a black-body surface	hr0		W/(m2·K)		X
	convective coefficient; internal surface	hc;i	h_ci_eli	W/(m2·K)		√
	convective coefficient; external surface	hc;e	h_ce_eli	W/(m2·K)		√
	radiative coefficient; internal surface	hr;i	h_ri_eli	W/(m2·K)		√

	radiative coefficient; external surface	hr;e	h_re_eli	W/(m2·K)		√
	hemispherical emissivity of surface	ε				X

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
52010:2017	Timezone	TZ	timezoneW			√
	latitude	φw	latitude_deg,	decimal degrees		√
	longitude	λw	longitude_deg,	decimal degrees		√
	Day of the year	nday	n_day	–		√
	Actual (clock) hour for the location (counting number of the hour in the day)	nhour	n_hour	–		√
	Diffuse solar irradiance on a horizontal surface	Gsol;d	DHI	W/m2		√
	Direct normal (beam) solar irradiance (normal to the sun)	Gsol;b	DNI	W/m2		√
	Solar constant	Gsol;c		W/m2		√
	Global solar irradiance	Gsol;g	GHI(?)	W/m2		√
	Ground solar reflectivity	ρsol;grnd	ground_solar_reflectivity	%		√
	Tilt angle of the inclined surface from horizontal, measured upwards facing	βic	beta_ic_deg	decimal degrees		√
	Orientation angle of the inclined surface, expressed as the geographical azimuth angle of the horizontal projection of the inclined surface normal d	γic	gamma_ic_deg	decimal degrees		√
	Base height of the shaded surface, from ground level	H0;ic		m		
	Height of the shaded surface, from bottom to top; if tilted: vertical projection d	H1;ic		m		
	Height of the shading obstacle, from ground level	Hsh;obst		m		
	Horizontal distance between the shaded surface and the shading object (obstacle) in the direction of the sun, measured between their central points	Lsh;obst		m		
	The position of the shading object indicated by the upper boundary of the geographical azimuth angle e	γsh;obst;max		°		
	Constant parameter for clearness formula	K		rad-3		
	Luminous efficacy	Kv		lm/W		

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	IV	oCE
52016-1	Object type (more than one choice possible)	EPB_OBJECT_TYPE		n/a	√	√
	Building category	BLDNG-CAT_TYPE	building_type	n/a	√	√
	Space category for each space or group of spaces (if different from building category)	SPACE-CAT_TYPE		n/a	√	√
	Application type	EPB_APPLIC_TYPE		n/a	√	√
	Assessment type	EPB_ASSESS_TYPE		n/a	√	√
	Calculation case	CASE_IDENTIFIER		n/a	√	√
	Useful floor area per elementary space	*A*use;*sp*	a_use	m2	√	√
	Air volume per elementary space	*V*int;a;*sp*	volume	m3	√	√
	Value of areal thermal capacity of air and furniture per thermally conditioned zone	*κ*m;int	km_int_a_zt	J/(m2·K)	√	√
	Class of the internal heat capacity per thermally conditioned zone	-	construction_class		√	√
	Thermal resistance, per opaque building element	*R*c;*k*	Rc_eli	(m2·K)/W	√	√
	Effective thermal resistance, per building element in thermal contact with the ground, including slab-on-ground floors, suspended floors and basements (including the effect of the ground)	*R*c;fl;eff;*k*	R_eff	(m2·K)/W	√	√
	Thermal resistance of a 0,5 m thick ground layer, per building element in thermal contact with the ground	*R*gr;k	R_gr	(m2·K)/W	√	√

	Thermal capacity of a 0,5 m thick ground layer, per building element in thermal contact with the ground	*κ*gr;k	kappa_gr	J/(m2·K)	√	√
	Thermal resistance of a virtual ground layer, per building element in thermal contact with the ground	*R*gr;vi;*k*	R_gr_ve	(m2·K)/W	√	√
	Virtual ground temperature, per building element in thermal contact with the ground, per month	*θ*gr;vi;*k*;*m*		°C	√	√
	Convective heat transfer coefficient internal surface, per construction element	*h*ci;*k*		W/(m2·K)	√	√
	Long-wave radiative heat transfer coefficient internal surface, per construction element	*h*ri;*k*		W/(m2·K)	√	√
	Class for the distribution of the mass, per opaque building element	-		-	√	√
	Class for the areal heat capacity, per opaque element	-		-	√	√
	Thermal transmittance, per opaque element	*U*c;op;*k*	U_eli	W/(m2·K)	√	√
	Ground transmission heat transfer coefficient for building elements in thermal contact with the ground, including slab-on-ground floors, suspended floors and basements, per thermal zone and month, based on the annual temperature difference	*H*gr;an;*ztc*;*m*		W/K		
	Average overall heat transfer coefficient for transmission through the ground floor, adjusted for the seasonal temperature difference, for the heating season, per thermal zone	*H*gr;H;adj;*ztc*		W/K		
	Average overall heat transfer coefficient for transmission through the ground floor, adjusted for the seasonal temperature difference, for the cooling season, per thermal zone	*H*gr;C;adj;*ztc*		W/K		
	Area, per construction element; in case of protruded components also the projected area	*A*el;*k*	area_elements	m2		
	Convective heat transfer coefficient external surface, per construction element	*h*ce;*k*	h_ce_eli	W/(m2·K)	√	√
	Long-wave radiative heat transfer coefficient external surface, per construction element	*h*re;*k*	h_re_eli	W/(m2·K)	√	√
	Thermal transmittance, per window	*U*w;*k*	U_eli	W/(m2·K)	√	√
	Thermal transmittance, per window with closed shutter	*U*wsht;*k*	U_eli	W/(m2·K)	√	√
	Thermal transmittance, per door	*U*d;*k*	U_eli	W/(m2·K)	√	√
	Thermal transmittance, per curtain wall	*U*cw;*k*	U_eli	W/(m2·K)	√	√
	Glazed area, per window element	*A*gl;*k*		m2		
	Linear thermal transmittance, per linear thermal bridge	*Ψ*tb;*k*		W/(m·K)		
	Length, per linear thermal bridge	*l*tb;*k*		m		
	Overall heat transfer coefficient for the thermal bridges, per thermal zone	*H*tr;tb;*zt*	thermal_bridge_heat	W/K	√	√
	Heat transfer coefficient between thermally conditioned zone and thermally unconditioned zone, per month	*H*ztc,*j*;*ztu*;*m*		W/K		
	Heat transfer coefficient between thermally unconditioned zone *ztu* and external environment, per month	*H*ztu;e;*m*		W/K		
	Heat transfer coefficient between thermally unconditioned zone *ztu* and external environment by transmission, per month	*H*tr;*ztu*;*e*;*m*		W/K		
	Tilt angle, per external building element (from horizontal, measured upwards facing)	*β*ic;*k*	beta_ic_deg	°	√	√
	Orientation angle per external building element, (expressed as the geographical azimuth angle of the horizontal projection of the inclined surface normal; convention: angle from South, eastwards positive, westwards negative)	*γ*ic;*k*	gamma_ic_deg	°	√	√
	Class for the solar absorption coefficient of the external surface, per opaque element	*a*sol;*k*	a_sol_pli_eli	-	√	√
	Total solar energy transmittance at normal incidence, for the transparent part, per transparent building element with non-scattering glazing	*g*gl;n;*k*	CorrectionWeighting-FactorGlazingB22b	-	√	√
	Total solar energy transmittance at 45 degrees (x=45) incidence and diffuse (x = dif) radiation, for the transparent part, per transparent building element with scattering glazing or blinds	*g*gl;x;*k*		-		
	Total solar energy transmittance including solar protection device of the transparent part, per transparent building element	*g*gl;sh;*k*		-		
	Heat transfer coefficient by transmission between zones *z* and *y*	*H*tr,*zy*		W/K		
	Ventilation heat transfer coefficient from zone *z* to zone *y*	*H*ve,*z*->*y*		W/K		
	Ventilation heat transfer coefficient from zone *y* to zone *z*	*H*ve,*y*->*z*		W/K		

	Thermal transmittance, per dynamic window or façade, per state	*U*dyn;*k;i*		W/(m2·K)		
	Total solar energy transmittance, per dynamic window or façade, per state *i*	*g*dyn;*k;i*		-		
	Solar transmittance, per dynamic window or façade, per state	*τ*sol; dyn;*k;i*		-		
	Visual transmittance, per dynamic window or façade, per state	*τ*vis; dyn;*k;i*		-		
	Latitude of the weather station	*φ*w		°		
	Base height, per shaded surface, from ground level	*H*0;ic;*k*		m		
	Height, per shaded surface, from bottom to top; if tilted: vertical projectione	*H*ic;*k*		m		
	Width, per shaded surface	*W*ic;*k*		m		
	Tilt angle, per external building element (from horizontal, measured upwards facing) e	*β*ic;*k*		°		
	Orientation angle per external building element, (expressed as the geographical azimuth angle of the horizontal projection of the inclined surface normal; convention: angle from South, eastwards positive, westwards negative) e	*γ*ic;*k*		°		
	Depth of a (simple) overhang (or similar shading object)	*Dk*;ovh,*q*		m		
	Vertical distance between the edge of the façade element and a (simple) overhang (or similar shading object)	*Lk*;ovh,*q*		m		
	Depth of a (simple) right side fin (or similar shading object)	*Dk*;finr,*r*		m		
	Horizontal distance between the edge of the façade element and a (simple) right side fin (or similar shading object)	*Lk*;finr,*r*		m		
	Depth of a (simple) left side fin (or similar shading object)	*Dk*;finl,*l*		m		
	Horizontal distance between the edge of the façade element and a (simple) left side fin (or similar shading object)	*Lk*;finl,*l*		m		
	Height from ground level, per shading obstacle in a skyline segment	*Hk*;obst;*p;i*		m		
	Horizontal distance between the shaded surface *k* and the shading object (obstacle) in a skyline segment, per shading object, measured between their central points	*Lk*;obst;*p;i*		m		
	Lowest height, from ground level, per shading overhang in a skyline segment, from ground level	*Hk*;ovh;*q;i*		m		
	Horizontal distance between the shaded surface *k* and the overhang in a skyline segment, per shading object, measured between their central points	*Lk*;ovh;*q;i*		m		
	Position, per skyline segment, indicated by the upper boundary of the geographical azimuth angle (convention: angle from South, eastwards positive, westwards negative)	*γ*sh;obst;max;*i*		°		
	Temperature set-point for heating, per thermally conditioned zone, per hour, taking into account intermittent heating (day, night, weekend) and unoccupied periods, if applicable	*θ*int;set;H;*ztc;t*	heating_setpoint	°C	√	√
	Temperature set-point for cooling, per thermally conditioned zone, per hour, taking into account intermittent cooling (e.g. weekend) and unoccupied periods, if applicable	*θ*int;set;C;*ztc;t*	cooling_setpoint	°C	√	√
	Available heating power, per thermally conditioned zone, per hour	*Φ*H;avail;*ztc;t*	power_heating_max	W	√	√
	Available cooling power, per thermally conditioned zone, per hour	*Φ*C;avail;*ztc;t*	power_cooling_max	W	√	√
	Convective fraction of the heating system per thermally conditioned zone (if system specific)	*f*H,c *ztc*	f_H_c	-	√	√
	Convective fraction of the cooling system per thermally conditioned zone (if system specific)	*f*C,c *ztc*	f_C_c	-	√	√
	Set-point relative humidity for humidification, per thermally conditioned zone, per hour	*φ*int;set;HU;*ztc;t*	phi_int_set_HU_t	%	√	√
	Set-point relative humidity for dehumidification, per thermally conditioned zone, per hour	*φ*int;set;DHU;*ztc;t*	phi_int_set_DHU_t	%	√	√
	Volume flow rate for each air flow element *k* entering a thermal zone, per hour	*q*V,k;t	air_flow_rate/ q_v_k_t	m3/s	√	√
	Supply temperature for each ventilation system air flow element *k* entering a zone, per hour	*θ*sup;*k;t*	theta_sup	°C	√	√
	Supply air moisture content for air flow element *k* entering a thermal zone, per hour	*x*a;sup;k;*t*	x_a_sup_k_t	kg/kg dry air	√	√
	Actual system specific moisture content of the mechanical supply air entering a zone, in case of central (de-)humidification of the mechanical ventilation, per thermally conditioned zone, per hour	*x*a;sup;ss;*zt;t*	x_a_sup_ss_t	kg/kg dry air	√	√
	Actual system specific moisture supply by the local system for humidification, per thermally conditioned zone, per hour	*G*HU;ss;*ztc;t*	G_HU_ss_ztc_t	kg/s	√	√
	Actual system specific moisture removal by the local system for dehumidification, per thermally conditioned zone. per hour	*G*DHU;ss;*ztc;t*	G_DHU_ss_ztc_t	kg/s	√	√
	The normal ('thermal comfort level') heating temperature set-point per thermally conditioned zone	*θ*int;set;H;*ztc*	heating_setpoint	°C	√	√
	The reduced ('economy level') heating temperature set-point of the zone (day, night and/or weekend)	*θ*int;set;H;low;*ztc*		°C		
	The duration of the period with reduced heating set-point (day, night and/or weekend)	Δ*t*H;red;y;*ztc*		h		
	The number of repetitions in a week of heating reduction period (day, night and/or weekend)	*n*rep;H;red;y;*ztc*		-		
	The normal ('thermal comfort level') cooling temperature set-point per thermally conditioned zone	*θ*int;set;C;*ztc*	cooling_setpoint	°C	√	√

	The number of hours during the weekend with reduced temperature set-point for cooling or interruption	ΔtC;red;wknd;ztc		-		
	The fraction of the month which is the unoccupied (heating/cooling) period	fH/C;nocc;ztc;m		-		
	Monthly time-average airflow rate of air flow element, k entering the thermal zone, for heating/cooling, for each month	qV;k;H/C;m	qvkt	m3/s	√	√
	Monthly mean supply temperature for each ventilation system air flow element, for heating/cooling, entering a zone, for each month	θsup;k;H/C;m	Theta_sup	°C	√	√
	The fraction of sensible energy need to be added for dehumidification, per type of cooling system	fDHU;C;ss		-		
	External (outdoor) air temperature at each hour	θe;a;t	Theta_ext	°C	√	√
	Moisture content or mixing ratio of the external air per hour	xa;e;t	x_a_e	kg/kg dry air	√	√
	Direct part (including circumsolar) of the solar irradiance, per building element, per hour	Isol;dir;tot;k;t	I_dir_tot	W/m2	√	√
	Diffuse part (excluding circumsolar; including ground reflection) of the solar irradiance, per building element, per hour	Isol;dif;tot;k;t	I_dif_tot	W/m2	√	√
	Solar altitude angle, from horizontal, per hour	αsol;t		°		
	Solar azimuth angle, per hour	φsol;t		°		
	(Convention in this document: angle from South, eastwards positive, westwards negative)					
	Duration, per month	Δtm		h		
	Mean external (outdoor) air temperature, per month	θe;a;m		°C		
	Mean external (outdoor) air temperature, per year	θe;a;an		°C		
	Total solar irradiation per building element, per month	Hsol;k;m	tau_sol_eli	kWh/m2	√	√
	Specific internal heat flow rate due to occupants, per thermal zone, per hour	$q$int;oc;zt;t	q_int_occ	W/m2	√	√
	Specific internal heat flow rate due to appliances, per thermal zone, per hour	$q$int;A;zt;t	q_int_a	W/m2	√	√
	Specific internal heat flow rate due to lighting, per thermal zone, per hour	$q$int;L;zt;t	q_int_L	W/m2	√	√
	Specific internal heat flow rate due to hot and mains water and sewage systems, per thermal zone, per hour	$q$int;WA;zt;t	q_int_WA	W/m2	√	√
	Specific internal heat flow rate due to HVAC, per thermal zone, per hour	$q$int;HVAC;zt;t	q_int_HVAC	W/m2	√	√
	Specific internal heat flow rate due to processes and goods, per thermal zone, per hour	$q$int;proc;zt;t	q_int_proc	W/m2	√	√
	Moisture production in the zone, per thermally conditioned zone, per hour	Gint;ztc;t	G_int_ztc_t	kg/s	√	√
	Specific internal heat gain due to occupants, for heating/cooling, per thermal zone, per month	QH/C;spec;int;oc;zt;m		W/m2		
	Specific internal heat gain due to appliances, for heating/cooling, per thermal zone, per month	QH/C;spec;int;A;zt;m		W/m2		
	Specific internal heat gain due to lighting, for heating/cooling, per thermal zone, per month	QH/C;spec;int;L;zt;m		W/m2		
	Specific internal heat gain due to hot and mains water and sewage systems, for heating/cooling, per thermal zone, per month	QH/C;spec;int;WA;zt;m		W/m2		
	Specific internal heat gain due to HVAC, for heating/cooling, per thermal zone, per month	QH/C;spec;int;HVAC;zt;m		W/m2		
	Specific internal heat gain due to processes and goods, for heating/cooling , per thermal zone, per month	QH/C;spec;int;proc;zt;m		W/m2		
	Atmospheric pressure	$p$atm	P_atm	Pa	√	√
	Specific heat of air at constant pressure	ca	Ca	J/(kg·K)	√	√
	Air density at 20 °C	ρa	rho_a_ref	kg/m3	√	√
	Latent heat of vaporization of water	$h$we	h_we	J/kg	√	√
	Stefan-Boltzmann constant	σ		W/(m2·K4)	√	√

7.2 ANNEX A.2 – Output Variables

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
52031-1	auxiliary energy — heating/cooling emission	Wem,ls,aux	Q_HC_em_aux	kWh	√
	additional energy losses of heat emission	Qem,ls	Q_em_ls	kWh	√
	equivalent internal heating temperature	θH;int;inc	theta_HC_int_inc	°C	√
	equivalent internal cooling temperature	θC;int;inc	theta_HC_int_inc	°C	√
	temperature variation based on losses	Δθint;inc	D_theta_int_inc_H / D_theta_int_inc_C	°C	√
	annual expenditure factor for the heat and cooling emission	emlsan,,			√
	convective fraction of the heating/cooling emitter	fem,conv			√

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
Heat Pump EPB Spread-sheet	EER for priority i	EERpi	eer_priority	-	√
	Required energy input for priority i	QC;gen,in;pi,hp	q_c_gen_in	kWh	√
	Energy rejected to the environment	QC,gen;rej;pi	q_c_gen_rej	kWh	√
	COP for priority i	COPpi	COP_final	-	√
	Required energy input for priority i	QH;gen,in;pi,hp	q_h_gen_in	kWh	√
	Energy captured from the environment	QH,gen;src;in;pi	q_h_gen_src_in	kWh	√

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
15316-4-1	Fuel heat input	Egen;in	E_gen_in	kWh	√
	Recoverable generation heat losses for heating system (in the calculation interval)	Qgen;ls;rbl	Q_gen_ls_rbl	kWh	√
	Generation heat losses (in the calculation interval)	Qgen;ls;	Q_gen_ls_rbl	kWh	√
	Expenditure factor of the generator for the whole service	εgen	E_gen	–	√
	Expenditure factor of the generator for heating	εH;gen	E_H_gen	–	√
	Expenditure factor of the generator for cooling	εC;gen	E_C_gen	–	√
	Expenditure factor of the generator for ventilation	εV;gen	E_V_gen	–	√
	Expenditure factor of the generator for DHW	εW;gen	E_W_gen	–	√
	Heat generation auxiliary energy for the heating system (in the calculation interval)	Wgen	W_gen	kWh	√
	Fuel type			List	√

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
52032-1/ EPB Spreadsheet	Thermal loss of the distribution system for heating in the zone	QH,dis,ls	Q_H_dis_ls_Hsa_tot	kWh	√
	Thermal loss of the distribution system for cooling in the zone	QC,dis,ls		kWh	
	Thermal loss of the distribution system for DHW in the zone	QW,dis,ls	Q_W_dloop_in_Wsa_j	kWh	√
	Recoverable thermal loss of the distribution system for heating in the zone	QH,dis,rbl	Q_H_dis_ls_rbl_Hsa_tot	kWh	√
	Recoverable thermal loss of the distribution system for cooling in the zone	QC,dis,rbl		kWh	
	Recoverable thermal loss of the distribution system for DHW in the zone	QW,dis,rbl	Q_W_dis_ls_rbl_Hsa_tot	kWh	√
	Auxiliary energy for distribution system heating in the zone	WH,dis	W_H_dis_aux_Hsa_j_m_i	kWh	√
	Auxiliary energy for distribution system cooling in the zone	WC,dis		kWh	
	Auxiliary energy for distribution system DHW in the zone	WW,dis	W_W_dloop_aux_Wsa_j	kWh	√
	Recoverable auxiliary energy for distribution system heating in the zone	QH,dis,rbl	Q_H_dis_ls_rbl_Hsa_tot	kWh	√
	Recoverable auxiliary energy for distribution system cooling in the zone	QC,dis,rbl		kWh	
	Recoverable auxiliary energy for distribution system DHW in the zone	QW,dis,rbl	Q_W_dloop_ls_rbl_Wsa_tot	kWh	√
	Recovered auxiliary energy for distribution system heating in the zone	QH,dis,rvd	Q_H_dis_aux_rvd_Hsa_j_m_i	kWh	√
	Recovered auxiliary energy for distribution system cooling in the zone	QC,dis,rvd	Q_C_dis_aux_rvd_Hsa_j_m_i	kWh	
	Recovered auxiliary energy for distribution system DHW in the zone	QW,dis,rvd	Q_W_dis_aux_rvd_Hsa	kWh	√

12831-3 / EPB Spreas-dheet	Effective power required for DHW	ΦW,eff		W	
	Storage volume of the hot water tank	Vsto		m³	
	Energy needs for DHW in the time step *i*	QW,nd,i	Q_W_nd_d	kWh/t	√
	Volume of water drawn at the current time step	Vt	V_W_nd_d	m³/ti	

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
EN 15316-5 EPB spreadsheet	Energy input for the storage from the generation system	QH;sto;X;in	Q_W_sto_in_tot	kWh	√
	Heat losses (recoverable)	QH;sto;ls;rbl	Q_sto_ls_vol	kWh	√
	Heat losses (non-recoverable)	QH;sto;ls;nrbl	-	kWh	
	Total auxiliary energy	WH;sto;aux	-	kWh	
	Temperature input from generation	ϑsto;gen;flw		°C	
	Temperature output to generation	ϑsto;gen;re		°C	
	Temperature (s) of the volume (s) of the storage unit	ϑsto;V,i	theta_sto_ini_vol	°C	√
	Temperature at the output of the heat exchanger (return to generation)	ϑsto;gen;dist;flw,i		°C	

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
13370	Thermal transmittance of slab on ground floor, including the effect of the ground	Ufg,sog	U_sog	W/(m²·K)	√
	Thermal transmittance of suspended floor, including the effect of the ground	Ufg,sus		W/(m²·K)	
	Thermal transmittance of wall of heated basement, including the effect of the ground	Uwg,b		W/(m²·K)	
	Thermal transmittance of floor of heated basement, including the effect of the ground	Ufg,b		W/(m²·K)	
	Effective thermal transmittance of whole heated basement, including the effect of the ground	Ubg,eff		W/(m²·K)	
	Thermal transmittance of unheated basement, including the effect of the ground	Uub		W/(m²·K)	
	Thermal resistance of slab on ground floor	Rf,sog	thermal_resistance_floor	m²·K/W	√
	Thermal resistance of suspended floor	Rf,sus		m²·K/W	
	Thermal resistance of basement wall	Rw,b		m²·K/W	
	Thermal resistance of basement floor	Rf,b		m²·K/W	
	Effective thermal resistance of floor construction (including the effect of the ground)	Rf,eff	total_thermal_transmittance	m²·K/W	√
	Ground heat transfer coefficient for each month	Hg,an,m	periodic_heat_flow_due_to_internal_temperature variation	W/K	√
	Ground heat transfer coefficient for the heating season	Hg,H,adj	periodic_heat_flow_due_to_internal_temperature variation	W/K	√
	Ground heat transfer coefficient for the cooling season	Hg,C,adj	periodic_heat_flow_due_to_internal_temperature variation	W/K	√
	Thermal resistance of 0.5 m of ground (for dynamic calculations according to Annex F)	Rg	R_gr	m²·K/W	√
	Thermal capacity of 0.5 m of ground (for dynamic calculations according to Annex F)	κg	kappa_pli_eli	J/(m²·K)	√
	Thermal resistance of virtual layer (for dynamic calculations according to Annex F)	Rvi	R_gr_ve	m²·K/W	√
	Virtual temperature in ground for each month (for dynamic calculations according to Annex F)	θvi		°C	√

Stand-ard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
52010:2017	Day of the year	nday	n_day	–	√
	Actual (clock) hour for the location (counting number of the hour in the day)	nhour	n_hour	–	√
	Calculated diffuse solar irradiance	Idif	I_dif	W/m2	√
	Calculated diffuse solar irradiance excluding circumsolar irradiance	Idif;tot	I_dif_tot	W/m2	√
	Calculated direct solar irradiance	Idir	I_dir	W/m2	√
	Calculated direct solar irradiance including circumsolar irradiance	Idir;tot	I_dir_tot	W/m2	√
	Calculated hemispherical solar irradiance	Itot	I_tots	W/m2	√
	Calculated global illuminance	Ev		lx	
	Calculated hemispherical solar irradiance, including effect of shading	Itot;sh		W/m2	
	Solar altitude angle, from horizontal	αsol	a_sol	°	√
	Solar azimuth angled	φsol	or_tilt_azim_dic	°	√
	Dry-bulb air temperature	θa	theta_ext	°C	√
	Wind speed at 10 m height	u10		m/s	√
	Wind direction from north	D		°	√

	Long-wave irradiance from the atmosphere on a horizontal plane	Gl;a		W/m2	
	Moisture content or mixing ratio	x	X	kg/kg	√
	Relative humidity	φ	RH%	%	

Standard	Variable Description	Symbol	Variable name in EPB Engine	Unit	oCE
52016-1:2017	Sensible energy need for heating, per thermally conditioned zone, per year	QH;nd;ztc;an	Q_H_ztc_annual	kWh	√
	Sensible energy need for cooling, per thermally conditioned zone, per year	QC;nd;ztc;an	Q_C_ztc_annual	kWh	√
	Sensible energy need for heating, per thermally conditioned zone, per month	QH;nd;ztc;m	QH_nd_ztc_monthly	kWh	√
	Sensible energy need for cooling, per thermally conditioned zone, per month	QC;nd;ztc;m	QC_nd_ztc_monthly	kWh	√
	Internal operative temperature, per thermally conditioned zone, per month	θint;op;ztc;m	Theta_op_ztc_monthly	°C	√
	Internal temperature, per thermally unconditioned zone, per month	θztu;m		°C	
	(Sensible) hourly heating load, per thermally conditioned zone, per hour	ΦH;ld;ztc;t	Phi_H_ztc	W	√
	(Sensible) hourly cooling load, per thermally conditioned zone	ΦC;ld;ztc;t	Phi_C_ztc	W	√
	Internal operative temperature, per thermally conditioned zone, per hour	θint;op;ztc;t	Theta_op_ztc	°C	√
	Internal mean radiant temperature, per thermally conditioned zone, per hour	θint;r;mn;ztc;t	Theta_mrt_ztc	°C	√
	Indoor air temperature, per thermally conditioned zone, per hour	θint;a;ztc;t	Theta_int_ztc	°C	√
	Humidification needs, per thermally conditioned zone, per year	QHU;nd;zt;an	Q_HU_ztc_annual	kWh	
	Dehumidification needs, per thermally conditioned zone, per year	QDHU;nd;zt;an	Q_DHU_ztc_annual	kWh	√
	Humidification needs, per thermally conditioned zone, per month	QHU;nd;zt;m	Q_HU_zt_monthly	kWh	√
	Dehumidification needs, per thermally conditioned zone, per month	QDHU;nd;zt;m	Q_DHU_zt_monthly	kWh	√
	Humidification moisture load, per thermally conditioned zone, per hour	GHU;ld;ztc;t	G_HU_ld_ztc	kg/s	√
	Dehumidification moisture (removal) load, per thermally conditioned zone, per hour	GDHU;ld;ztc;t	G_DHU_ld_ztc	kg/s	√
	Latent heat load for humidification (humidification load), per thermally conditioned zone, per hour	ΦHU;ld;zt;t	Phi_HU_ld_ztc	W	√
	Latent heat load for dehumidification (dehumidification load), per thermally conditioned zone, per hour	ΦDHU;ld;zt;t	Phi_DHU_ld_ztc	W	√
	Required moisture content for central humidification of the mechanical supply air, per thermally conditioned zone, per hour	xa;sup;HU;req;ztc;t	x_a_sup_HU_req_ztc	kg/kg dry air	√
	Required moisture content for central dehumidification of the mechanical supply air, per thermally conditioned zone, per hour	xa;sup;DHU;req;ztc;t	x_a_sup_DHU_req_ztc	kg/kg dry air	√
	Indoor moisture content, per thermally conditioned zone, per hour	xint;a;ztc;t	x_int_a_ztc	kg/kg dry air	√
	Design sensible heating load, per year	ΦH;ld;des;ztc;an	Phi_H_des_ztc		√
	Design sensible cooling load, per year	ΦC;ld;des;ztc;an	Phi_C_des_ztc		√
	Design humidification moisture load, per year	GHU;ld;des;ztc;an	G_HU_ld_des_ztc_annual		√
	Design dehumidification moisture load, per year	GDHU;ld;des;ztc;an	G_DHU_ld_des_ztc_annual		√
	Design latent heat load for humidification, per year	ΦHU;ld;des;ztc;an	Phi_HU_ld_des_ztc_annual		√
	Design latent heat load for dehumidification, per year	ΦDHU;ld;des;ztc;an	Phi_DHU_ld_des_ztc_annual		√
	Design increase of the supply air moisture content, compared to the moisture content of external air, for humidification, per year	Δxa;sup;HU;ld;des;ztc;an	Delta_x_a_sup_HU_des_ztc		√
	Design decrease of the supply air moisture content, compared to the moisture content of external air, for dehumidification, per year	Δxa;sup;DHU;ld;des;ztc;an	Delta_x_a_sup_DHU_des_ztc		√
	Annual amount of undersizing of the heating system for different thresholds *Φ*UH;ld;thres,*i*	QUH;thres,i,ztc;an		kWh	
	Annual amount of undersizing of the cooling system for different thresholds *Φ*UC;ld;thres,*i*	QUC;thres,i,ztc;an		kWh	
	Annual accumulated undertemperature (“underheating”) for different temperature differences Δ*θ*UH;thres,*i*	TUH;thres,i,ztc;an		K·h	
	Annual accumulated overtemperature for different temperature differences Δ*θ*OH;thres,*i*	TOH;thres,i,ztc;an		K·h	
	Monthly heat balance ratio, heating	γH;gn;ztc;m		-	
	Monthly heat balance ratio, cooling	γC;ht;ztc;m		-	
	Monthly utilization factor, heating	ηH;gn;ztc;m		-	
	Monthly utilization factor, cooling	ηC;ht;ztc;m		-	
	(annual) design sensible heating load in the subsystem	ΦH;ld;des;ss;an		W	
	(annual) design sensible cooling load in the subsystem	ΦC;ld;des;ss;an		W	