D2.3 – Integrated SRI for Enhanced Energy Performance Assessments and Financing Practices

Submission date:

Due date: 02/02/2026

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 D2.3**
Responsible Partner	**EUP**
WP and Task no. and title	**WP2 – T2.3 Integration of Smart Readiness Indicator (SRI) into EU Kernel and Financing Practices**
Version	**v.1**
Version Date	**02/02/2026**

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**
Author/s	**Tibet Bashkaya**
Task Leader	**EUP**
WP Leader	**QUE**
1st Reviewer	**Andrei Vladimir Litiu, EPBC**
2nd Reviewer	**Chloe Walsh, EU.BAC**

ABREVIATIONS

BEP	Building Energy Performance
EPC	Energy Performance Certificates
SRI	Smart Readiness Indicator
WPs	Work Packages
EPBD	Energy Performance of Buildings Directive

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**	European Building Automation and Controls Association
8	**EGC**	European Green Cities
9	**EFIEES**	European Federation of Intelligent Energy Efficiency Services

About openBEP4EU

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

The building sector is recognised as a cornerstone in achieving the European Union's climate neutrality target by 2050. In line with the EU's twin transition, encompassing both digital and green transformation, the sector plays a critical role in addressing energy consumption and emissions. Energy Performance Certificates (EPCs), alongside Smart Readiness Indicators (SRIs), have emerged as essential tools for driving the improvement of building performance. EPCs provide insight into energy efficiency, while SRIs assess the capacity of buildings to adapt through smart technologies that enhance performance, responsiveness and user interaction. Together, they enable a more comprehensive and actionable understanding of building functionality.

To support the harmonised implementation of these frameworks across Member States, the OpenBEP4EU project aims to deliver open-source, standards-based software tools: the openBEP4EU EPC calculation engine and the openBEP4EU SRI calculation engine. These tools are designed to facilitate consistent, high-quality assessments while ensuring interoperability and transparency. By enabling simultaneous adoption of innovative EPC and SRI methodologies, the project contributes to improved comparability, stakeholder trust, and the creation of new value streams around building performance data. This data will empower actors such as policymakers, energy service providers, building owners, and financial institutions to develop targeted policies, market incentives, and financing mechanisms to accelerate deep renovation and smart readiness upgrades across the EU.

Executive Summary

This deliverable presents the development and integration of the Smart Readiness Indicator (SRI) Kernel Engine within the EU Kernel Energy Performance Certificate (EPC) architecture, implemented under Task 2.3. The goal is to operate the European SRI approach in a standardized, automated, and interoperable way so that it can be systematically incorporated into digital EPC workflows and used in applications linked to financing and rehabilitation planning.

The official European SRI calculation framework is fully implemented by the SRI Kernel Engine, a stand-alone, cross-platform computational component that complies with the Energy Performance of Buildings Directive (EPBD) and associated requirements. The kernel provides uniform validation, calculation, conversion, and comparison of SRI assessments using standardized JSON and XML data formats by converting the well-established spreadsheet-based approach into a machine-executable engine. The SRI Kernel is intended to be used in harmonized EPC assessment pipelines and is included in the EU Kernel EPC Engine. It generates structured SRI outputs, including overall, impact-level, domain-level, and aggregated scores aligned with EPBD key functionalities, and processes structured building and assessment data from heterogeneous sources. Strict validation rules are applied to ensure methodological compliance and data completeness [1] [2]

Through predetermined weighting methods, the kernel-based integration permits controlled adaptability to climatic and building-specific conditions while ensuring uniform execution of SRI assessments across tools and national contexts, promoting cross-country harmonization. The comparability and reliability of SRI outcomes across Member States are increased, data quality is improved, and implementation complexity is decreased by centralizing SRI logic within a single authoritative engine

TABLE OF CONTENTS

ABOUT OPENBEP4EU **3**

EXECUTIVE SUMMARY **4**

TABLE OF CONTENTS **5**

LIST OF FIGURES 6

1. INTRODUCTION **7**

1.1. PURPOSE AND SCOPE 7
1.2. SRI INTEGRATION WITHIN THE OPENBEP4EU FRAMEWORK 7
1.3. LINK WITH EU LEGISLATIONS 8

2. CONCEPTUAL FRAMEWORK **9**

2.1. SMART READINESS INDICATOR (SRI): OVERVIEW & METHODOLOGY AND KERNEL ADAPTATION 9
2.2. SRI ASSESSMENT PROCESS 13

3. METHODOLOGY **14**

3.1. SRI KERNEL INTERNAL ARCHITECTURE 14
3.2. BUILD, PUBLISH AND COMMAND-LINE USAGE 17

4. DATA STRUCTURE AND REQUIREMENTS **21**

4.1. INPUT DATA SPECIFICATIONS (XML, JSON) 21
4.2. OUTPUT DATA SPECIFICATIONS (XML) 26

5. IMPLEMENTATION CHALLENGES & MITIGATION STRATEGIES **30**

5.1. TECHNICAL CHALLENGES 30
5.2. DATA AVAILABILITY AND STANDARDISATION 31
5.3. CROSS-COUNTRY HARMONISATION 31

6. CONCLUSION & NEXT STEPS **33**

6.1. CONCLUSION 33
6.2. NEXT STEPS 33

7. REFERENCES **35**

LIST OF FIGURES

Figure 1 - Seven Impact Categories of SRI [4] 9
Figure 2 - Overview of the Scoring Matrix, Containing the 7 Impact Criteria and 9 Technical Domains [4] 10
Figure 3 - Directory Structure of SRI Kernel 14
Figure 4 - Building the SRI Kernel project using the dotnet CLI 17
Figure 5 - Publishing the SRI Kernel application as a single, self -contained executable for multiple platforms. 17
Figure 6 - General Syntax for Invoking the "validate" operation 18
Figure 7 - Example usage of the "validate" operation with a JSON building data file 18
Figure 8 - General Syntax for invoking the “calculate" operation 19
Figure 9 - Example usage of the "calculate" operation, producing SRI scores from an XML input file 19
Figure 10 - General syntax for invoking the “convert" operation 19
Figure 11 - Example usage of the "convert" operation, converting building data from XML to JSON 19
Figure 12 - General syntax for invoking the "compare" operation 20
Figure 13 - Example usage of the "compare" operation, comparing two SRI score files in different formats 20
Figure 14 - Service Codes and names for Method A 21
Figure 15 - Service Codes and Method Names for Method B 22
Figure 16 - Indicative JSON structure of a "BuildingData" input file used by the SRI Kernel 23
Figure 17 - XML structure of a “BuildingData” input file used by SRI Kernel 25
Figure 18 - Indicative JSON structure of a "ScoresData" input file used by the SRI Kernel 27
Figure 19 - Indicative XML Structure of a "ScoresData" input file used by SRI Kernel 28

1. Introduction

1.1. Purpose and Scope

The deliverable focuses on the methodical integration of the Smart Readiness Indicator (SRI) into EU Kernel EPC processes and related finance practices, showcases the effective creation and customization of the SRI Kernel Engine within the framework of Task 2.3. SRI evaluations may be validated, calculated, converted, and compared using standardized data formats thanks to the SRI Kernel, automation-ready computational engine. The Kernel makes it possible to use SRI as a supplementary eligibility and readiness indicator for creative financing schemes supporting energy-efficient building renovations and improves the quality, reliability, and comparability of energy performance assessments by producing robust and interoperable SRI outputs.

The deliverable details integration-ready SRI Kernel Engine and how it fits into the EU Kernel EPC Engine. The deliverable shows how SRI computations are operationalized in a standardized, automated, and quality-assured way to enable downstream financial applications, digital tools, and EPC workflows. Technical architecture, data needs, integration technique, and implementation considerations are all included in the scope.

1.2. SRI Integration within the OpenBEP4EU Framework

The inclusion of the Smart Readiness Indicator (SRI) is intended to be a key facilitator for improving the market relevance, intelligence, and interoperability of Energy Performance Certificates (EPCs) inside the OpenBEP4EU framework. In order to ensure that smart readiness evaluations are smoothly integrated into digital EPC workflows in a standardized, scalable, and automation-ready way, the SRI Kernel created is positioned as a dedicated computational component of the EU Kernel EPC Engine.

The EU Kernel can use programmatic interfaces or command-line execution to access the SRI Kernel, which functions as a separate but completely compatible module. In addition to energy performance calculations, data validation procedures, and reporting services, this design enables the EU Kernel to coordinate SRI computations as part of larger EPC assessment pipelines. The SRI Kernel supports the OpenBEP4EU goal of facilitating interoperable, service-oriented building performance ecosystems by utilizing structured and machine-readable input and output formats (JSON and XML).

EPC tools, building audit platforms, digital twin environments, and supported data collection procedures like contact centres are the sources of the building data inputs. To guarantee methodological consistency, domain and service information completeness, and adherence to SRI methodological requirements, the kernel implements stringent validation rules prior to computation. The robustness and dependability of SRI results included in EPC outputs are greatly enhanced by this validation phase.

The EU Kernel EPC Engine receives the findings generated by the SRI Kernel in an organized format that facilitates storage, comparison, visualization, and further processing. This allows OpenBEP4EU services to input aggregated datasets into monitoring and benchmarking tools at the regional, national, or EU level, facilitate longitudinal analysis (e.g., before and after refurbishment), and provide smart preparedness findings alongside standard EPC metrics.

Additionally, the EU Kernel may make smart preparedness indicators available to external stakeholders like financing institutions, one-stop shops, and refurbishment facilitators by providing standardized, validated, and comparable SRI outputs. In this regard, SRI scores can serve as supplementary indications for evaluating digital maturity, renovation readiness, and the potential value of smart technology, supporting the creation of creative financing options in line with energy-efficient and smart building improvements.

1.3. Link with EU Legislations

By making organized, repeatable, and transparent SRI assessments possible, the SRI Kernel facilitates the execution of EU regulatory frameworks pertaining to smart preparedness and building performance. Its standardized input-output paradigm supports policy monitoring and alignment with EU-level procedures by facilitating harmonized reporting and quality control. The OpenBEP4EU framework's implementation of the Smart Readiness Indicator (SRI) is in line with the changing European regulatory environment that oversees building digitalisation and energy performance. The SRI Kernel, which was created as part of Task 2.3, offers a practical technical method for carrying out EU legal requirements for data harmonization, smart readiness, and the digital transformation of Energy Performance Certificates (EPCs).

The general framework for evaluating and enhancing the energy performance of buildings within the European Union is established by the Energy Performance of Buildings Directive (EPBD) [3]. The EU's realization that building performance must progressively take digital control, automation, and interaction with energy systems into consideration is reflected in the introduction of the Smart Readiness Indicator inside the EPBD.

In this regard, by offering a standardized and automated calculation engine that can be incorporated into EPC processes, the SRI Kernel operationalizes the EPBD's smart ready provisions. The kernel helps Member States implement EPBD requirements in a uniform and repeatable way by guaranteeing consistent application of SRI methodology and structured generation of SRI outputs. The EU Kernel EPC Engine's integration guarantees that SRI evaluations are integrated with conventional energy performance metrics, strengthening EPCs as all-encompassing tools for assessing building performance.

The methodological ideas outlined in this directive, such as the structured evaluation of building services, domains, and impacts, are applied in the creation of the SRI Kernel. To guarantee methodological compliance, the kernel applies stringent validation requirements to input data, including consistency in weighting and compliance values, adherence to calculation methodologies (Method A or Method B), and completeness of domain and service coverage. The SRI Kernel supports the consistent implementation of Directive (EU) (EU/2024/1275, EPBD) across various tools and national settings and lowers the risk of non-compliant or inconsistent evaluations by including the directive into the calculation process [1] [2] [3].

The digitalization of EPCs and the development of interoperable building data spaces are highlighted in recent EU policy initiatives [3]. By enabling automated SRI assessment processing inside digital workflows and enforcing defined data models appropriate for inclusion into EPC registries and digital building platforms, the SRI Kernel supports these goals. Data quality, traceability, and reuse all crucial components of EU data governance strategies that are supported by the kernel's capacity to validate, transform, and compare SRI datasets. These features help policy monitoring, benchmarking, and ongoing assessment methodology improvement by making it easier to save and analyze SRI data over an extended period of time.

The integration of SRI into OpenBEP4EU promotes EU goals pertaining to sustainable finance and rehabilitation options in addition to building performance directive. The kernel makes it possible to employ smart readiness measurements as complementary indicators for evaluating renovation potential, digital maturity, and system adaptability by providing consistent and trustworthy SRI outputs. The synchronization of technical building assessments with finance instruments, renovation incentives, and investment decision-making is encouraged by evolving EU policy objectives.

2. Conceptual Framework

2.1. Smart Readiness Indicator (SRI): Overview & Methodology and Kernel Adaptation

The framework serves as the standard implementation for SRI testing and deployment throughout Member States and is built using the multi-criteria assessment methodology set forth by the Commission Delegated Regulation (EU) 2020/2155 and the modalities via Commission Implementing Regulation (EU) 2020/2156 [1] [2]. By evaluating the existence and operation of smart-ready services made possible by building automation, control, and digital technology, the SRI approach assesses a building's smart readiness. The technology-neutral definition of these services guarantees that the evaluation concentrates on functional results rather than particular goods or suppliers.

The calculation framework is structured around:

- **7 impact criteria**, grouped under three high-level EPBD key functionalities [4]:
    - *Optimisation of energy efficiency and overall, in-use performance* (energy savings, maintenance and fault prediction)
    - *Adaptation to the needs of the occupant* (comfort, convenience, health and well-being, information to occupants)
    - *Adaptation to signals from the grid* (energy flexibility and storage)

This structure is illustrated in the official scoring matrix combining domains and impact criteria, which forms the backbone of the SRI calculation logic implemented in the SRI Kernel.

Figure 1 - Seven Impact Categories of SRI [4]

These three high-level EPBD functionalities are decomposed into **seven impact criteria**, as illustrated in the official SRI impact-category figure 1(Figure 1“Seven impact categories of the SRI”):

- Energy savings
- Maintenance and fault prediction

- Comfort
- Convenience
- Health, well-being and accessibility
- Information to occupants
- Energy flexibility and storage

Furthermore, these seven impact criteria are branched **into 9 technical domains** (e.g. heating, cooling, domestic hot water, ventilation, lighting, dynamic building envelope, electricity, electric vehicle charging, monitoring and control).

The interaction between the **7 impact criteria** and the **9 technical domains** is represented by the official **SRI scoring matrix** (Figure 2) Practical Guide SRI calculation. Each cell of this matrix reflects the contribution of a given domain to a specific impact criterion [4].

	Overall SRI score (%) + SRI class						
	%		%				%
	Optimise energy efficiency and overall in-use performance		Adapt its operation to the needs of the occupant				Adapt to signals from the grid (energy flexibility)
	%	%	%	%	%	%	%
	Energy efficiency	Maintenance and fault prediction	Comfort	Convenience	Health, well-being and accessibility	Information to occupants	Energy flexibility and storage
Heating	%	%	%	%	%	%	%
Cooling	%	%	%	%	%	%	%
Domestic hot water	%	%	%	%	%	%	%
Ventilation	%	%	%	%	%	%	%
Lighting	%	%	%	%	%	%	%
Dynamic building envelope	%	%	%	%	%	%	%
Electricity	%	%				%	%
Electric vehicle charging		%		%		%	%
Monitoring and control	%	%	%	%	%	%	%

Figure 2 - Overview of the Scoring Matrix, Containing the 7 Impact Criteria and 9 Technical Domains [4]

Kernel mapping:

In the SRI Kernel, this matrix is implicitly implemented through:

- service-level impact score tables embedded in the calculation logic
- aggregation rules that generate DomainScores, ImpactScores, and DetailedScores within ScoresData which is explained in Section 3.

2.1.1 Service-Based and Functionality-Level Assessment

Within each technical domain, the SRI framework defines a catalogue of smart-ready services. Each service can be implemented at different functionality levels, where higher levels represent more advanced, automated, or adaptive operation. The SRI methodology assumes that higher functionality levels generally deliver greater benefits across one or more impact criteria.

The SRI Kernel Engine operationalises this approach by accepting structured input data describing:

- which services are applicable to a given building
- the functionality level achieved by each service
- the share of the building to which a functionality level applies (allowing partial compliance).

This mirrors the logic of the official calculation spreadsheet, while enabling automated and reproducible execution within digital EPC workflows.

Kernel mapping:

This information is provided to the SRI Kernel via the BuildingData input structure, including:

- declared domains present or absent,
- selected assessment method (A or B),
- service applicability flags,
- functionality level values,
- optional partial-compliance shares.

2.1.2 Calculation Methods: Method A and Method B

The SRI Kernel supports both official assessment methods defined in the SRI framework:

- **Method A (simplified method)**, based on a reduced service catalogue and intended primarily for small or low complexity buildings
- **Method B (detailed method)**, based on a more comprehensive service catalogue and intended for larger or more complex buildings, particularly non-residential ones.

The framework of both approaches is similar, but technique A employs a smaller variety of services, necessitating less work and skill to complete the assessment. While the more detailed Method B is primarily focused on buildings with a higher complexity (usually large non-residential buildings, potentially large multi-family homes), the simplified Method A was anticipated to be primarily focused on small buildings with low complexity (single-family homes, small multi-family homes, small non-residential buildings, etc.). The weighting criteria suggested in the consolidated approach of the technical support studies have also been applied in the same calculation tool as methods A and B.

Kernel Mapping:

The selected method is explicitly declared in BuildingData.Method, which determines:

- the mandatory service list,

- applicable validation constraints,
- calculation paths inside the kernel.

2.1.3 Weighting, Normalisation, and Aggregation Logic

The SRI calculation is based on a **multi-level aggregation process**:

1. Service-level functionality scores are aggregated into **domain scores**
2. Domain scores are aggregated into **impact scores**, using predefined or user-defined weighting factors
3. Impact scores are further aggregated into a **single overall SRI score**, expressed as a percentage representing the ratio between the achieved smart readiness and the maximum obtainable smart readiness for the building.

The framework applies a mixed weighting approach that combines fixed weights, equal weights, and energy-balance-based weights depending on the impact criterion and technical domain. Default weighting factors are derived from European building stock data and climate zones, while allowing controlled adjustments where required by national implementation rules.

The SRI Kernel Engine faithfully implements this logic, ensuring that weighting consistency, normalisation rules, and methodological constraints are systematically enforced.

Kernel mapping:

The SRI Kernel:

- applies default weighting factors automatically,
- enforces sum-to-100% constraints,
- supports user-defined weighting scenarios where permitted,
- outputs all intermediate and final weighted results within ScoresData.

2.1.4 Output and Interpretation of Results

The result of the SRI calculation consists of:

- an overall SRI score
- impact-level scores for each of the seven impact criteria
- domain-level scores for each technical domain
- detailed scores combining domains and impacts in a matrix structure
- aggregated scores for the three EPBD key functionalities.

These outputs provide both high-level benchmarking information and detailed diagnostic insights, enabling their use in EPC enhancement, renovation planning, monitoring, and financing-related applications.

2.2. SRI Assessment Process

Despite the differences between methods A and B there is one common SRI calculation methodology. It can be described in seven steps, which we will discuss below.

2.2.1 Triage Process

The first step is the process of examining the building and documenting the present technical domains, as well as the present smart services included in these domains. As for the absent domains, you are given the option to select "absent but mandatory", which means that for this domain, although it is absent, all of its services will be taken into account at the maximum obtainable score. As for the present domains, in each smart service you are given the option of "not applicable". If enabled, it means that the smart service is absent, and next you can choose whether that service will "affect the maximum obtainable score or not". This is a process of examining whether a number or all of a domain's services are desired or ought to be present according to legislation. Note that according to the above, the following two scenarios are equivalent:

- *A domain is absent but mandatory.*
- *A domain is present and all of its services are not applicable and chosen to affect the maximum obtainable score.*

The triage process is time-consuming, especially for large, complex buildings, and plays a significant role in the next steps and the determination of the final results.

2.2.2. Functionality Level Evaluation

The triage process is followed by the services' functionality level assessment. Functionality levels vary from the value 0, when the examined service is deemed not functional at all, to the maximum value 4. For instance, the functionality level of a service in relation to heat emission control would be zero if there is no automatic control or would be graded with the maximum score if control utilizing sensors and information to the user is provided for each room of a building. This step may require in-site inspection, technical document study/provision, or a combination of them, depending on the building. The tool allows the use of multiple functionality levels for a specific smart service. This scenario is being interpreted as follows: the building has partial compliance with that functionality level of a smart service. This compliance must be specified in percentage of the total building area. For a service, the compliance percentages of the various functionality levels must sum up to 100%.

3. Methodology

3.1. SRI Kernel Internal Architecture

SRI Kernel is a cross-platform .NET command-line application providing a unified interface for:

- Validating “BuildingData” and “ScoresData”,
- Calculating Smart Readiness Indicator (SRI) scores using the external library SRICalculator.dll comprising the SRICalculator©,
- Producing output in multiple formats (either JSON or XML),
- Converting data between JSON and XML formats
- Comparing two input datasets (either “BuildingData” or “ScoresData”)

The SRI Kernel acts as a wrapper around the logic implemented in SRI Calculator, exposing high-level Command-Line Interface (CLI) functions for automated pipelines, integrations, and batch processing

3.1.1 Project Structure and Central Static API

The Kernel.cs file exposes the following public static operations. This is the central static Application Programming Interface (API) of the SRI Kernel. The tree structure of the directory that contains the source code of the SRI Kernel is as shown in Figure 3.

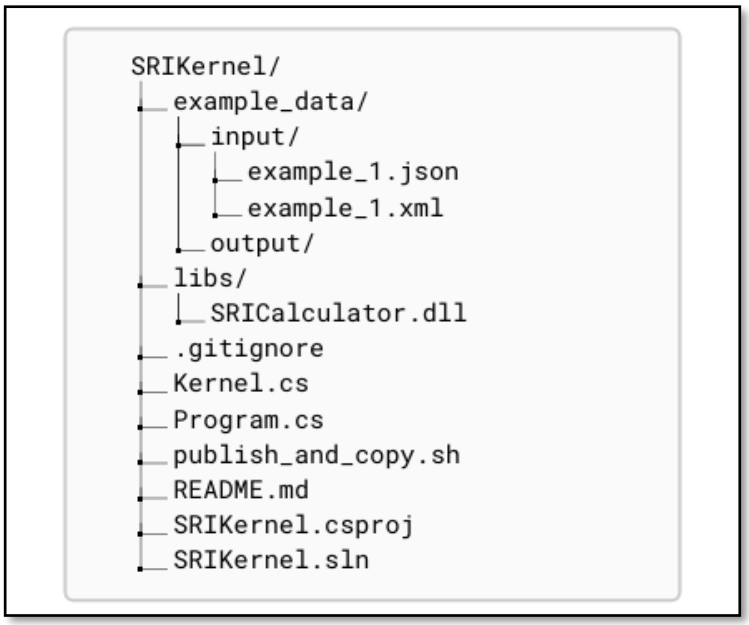

Figure 3 - Directory Structure of SRI Kernel

- **Method “Calculate”**

Method: Calculate (inputType, inputPath, outputType, outputPath)

Return type: void

Arguments:

- inputType: string (either “json” or “xml”)

- inputPath: string (i.e., the path of the input file)
- outputType: string (either “json” or “xml”)
- outputPath: string (i.e., the path that the output file will be written)

Description: This method does the following:

1. Validates the structure of the input file, which must represent a “BuildingData” object
2. Computes the SRI scores
3. Outputs a “ScoresData” object that is being written in a JSON or XML file

- **Method “ValidateBuildingData”**

Method: ValidateBuildingData(inputType, inputPath)

Return type: boolean

Arguments:

- inputType: string (either “json” or “xml”)
- inputPath: string (i.e., the path of the input file)

Description: This method validates the structure of a JSON or XML file against the structure of a “BuildingData” object

- **Method “ValidateScoresData”**

Method: ValidateScoresData(inputType, inputPath)

Return type: boolean

Arguments:

- inputType: string (either “json” or “xml”)
- inputPath: string (i.e., the path of the input file)

Description: This method validates the structure of a JSON or XML file against the structure of a “ScoresData” object.

- **Method “ConvertBuildingData”**

Method: ConvertBuildingData(inputType, inputPath, outputType, outputPath)

Return type: void

Arguments:

- inputType: string (either “json” or “xml”)
- inputPath: string (i.e., the path of the input file)
- outputType: string (either “json” or “xml”)
- outputPath: string (i.e., the path that the output file will be written)

Description: This method does the following:

1. Validates the structure of the input file representing a “BuildingData” object
2. Converts the input file between JSON and XML formats

- **Method “ConvertScoresData”**

Method: ConvertScoresData(inputType, inputPath, outputType, outputPath)

Return type: void

Arguments:

- inputType: string (either “json” or “xml”)
- inputPath: string (i.e., the path of the input file)
- outputType: string (either “json” or “xml”)
- outputPath: string (i.e., the path that the output file will be written)

Description: This method does the following:

1. validates the structure of the input file representing a “ScoresData” object
2. converts the input file between JSON and XML formats

- **Method “CompareBuildingData”**

Method: CompareBuildingData(inputType1, inputPath1, inputType2, inputPath2)

Return type: boolean

Arguments:

- inputType1: string (either “json” or “xml”)
- inputPath1: string (i.e., the path of the first input file)
- inputType2: string (either “json” or “xml”)
- inputPath2: string (i.e., the path of the second input file)

Description: This method does the following:

1. validates the structure of the two input files representing two “Building Data” objects
2. compares those files and decides whether they contain identical information or not

- **Method “CompareScoresData”**

Method: CompareScoresData(inputType1,inputPath1,inputType2,inputPath2)

Return type: boolean

Arguments:

- inputType1: string (either “json” or “xml”)
- inputPath1: string (i.e., the path of the first input file)
- inputType2: string (either “json” or “xml”)
- inputPath2: string (i.e., the path of the second input file)

Description: This method does the following:

1. validates the structure of the two input files representing two “ScoresData” objects
2. compares those files and decides whether they contain identical information or not

3.1.2 Command-Line Interface and Dependencies

- The Program.cs file interprets the user commands and offers the CLI. The supported operations are to validate, calculate, compare and convert. The CLI performs argument validation and delegates logic to the Kernel.cs,i.e the Central Static API.
- The SRI Kernel uses the external library SRICalculator.dll. That library contains the SRI Calculator. So, SRI Kernel depends on the following classes that are contained in the SRI Calculator:
  - "BuildingDataSerializer"
  - "ScoresDataSerializer"
  - "BuildingDataValidator"
  - ScoresDataValidator"
  - "Calculator"
- The SRI Kernel also depends on the following NuGet packages:
  - "MathNet.Numerics 5.0.0"

3.2. Build, Publish and Command-line Usage

3.2.1 Build

- To compile the SRI Kernel project, execute the command show in Figure 4 from the project root directory.

```
dotnet build
```

Figure 4 - Building the SRI Kernel project using the dotnet CLI

3.2.2 Publish

To publish the application for all the different platforms, you may choose one of the options shown below.

- **Option A**

To publish the application as a single, self-contained executable for multiple platforms, run the following commands. Each command produces a platform specific binary targeting linux-x64, osx-x64, and win-x64, respectively.

```
dotnet publish -c Release -r linux-x64 --self-contained
↪ true /p:PublishSingleFile=true
dotnet publish -c Release -r osx-x64   --self-contained
↪ true /p:PublishSingleFile=true
dotnet publish -c Release -r win-x64   --self-contained
↪ true /p:PublishSingleFile=true
```

Figure 5 - Publishing the SRI Kernel application as a single, self -contained executable for multiple platforms.

After execution, the generated binaries can be found at the following locations:

- bin/Release/net8.0/linux-x64/publish/SRIKernel
- bin/Release/net8.0/osx-x64/publish/SRIKernel
- bin/Release/net8.0/win-x64/publish/SRIKernel.exe

- **Option B**

To publish the application for all supported platforms in a single step, use the provided shell script publish_and_copy.sh. This approach automates the build process and produces self-contained, single-file executables for each target platform.

Before execution, ensure that:

- The host operating system is a Linux distribution or macOS.
- The script has execution permissions.

3.2.3 Command-Line Usage

As mentioned above, there are three executable files available for the different operating system types:

- win-x64: The executable file is the “SRIKernel.exe”.
- linux-x64: The executable file is the “SRIKernel”.
- osx-x64: The executable file is the “SRIKernel”.

Below, for the command-line usage and the examples, we will use the executable corresponding to linux-x64

- **Operation “validate”**

The “validate” operation checks the syntactic and semantic correctness of an input file against the expected SRI Kernel data model. The general command-line usage is shown in Figure 6.

```
SRIKernel validate <inputClass> <inputType> <inputPath>
```

Figure 6 - General Syntax for Invoking the "validate" operation

The command parameters are defined as follows:

- <inputClass>: the class of data to be validated (i.e., buildingdata or scoresdata)
- <inputType>: the input file format (i.e., json or xml)
- <inputPath>: the file-system path to the input file

An example invocation of the “validate” operation is shown in Figure 7.

```
SRIKernel validate buildingdata json example_1.json
```

Figure 7 - Example usage of the "validate" operation with a JSON building data file

- **Operation “calculate”**

The “calculate” operation executes the SRI computation for a given input file and produces the corresponding results in the specified output format. The general command-line syntax is shown in Figure 8.

```
SRIKernel calculate <inputType> <inputPath> <outputType>
↪ <outputPath>
```

Figure 8 - General Syntax for invoking the “calculate" operation

The command parameters are defined as follows:

- <inputType>: the input file format (i.e., json or xml)
- <inputPath>: the file-system path to the input file
- <outputType>: the output file format (i.e., json or xml)
- <outputPath>: the file-system path where the output file will be written

An example invocation of the “calculate” operation is shown in Figure 9.

```
SRIKernel calculate xml building.xml xml scores.xml
```

Figure 9 - Example usage of the "calculate" operation, producing SRI scores from an XML input file

- **Operation “convert”**

The “convert” operation transforms an input file between supported data formats while preserving its semantic content. It supports both building data and SRI score data and enables bidirectional conversion between JSON and XML representations. The general command-line syntax is shown in Figure 10.

The command parameters are defined as follows:

- <inputClass>: the class of data to be converted (i.e., buildingdata or scoresdata)

```
SRIKernel convert <inputClass> <inputType> <inputPath>
↪ <outputType> <outputPath>
```

Figure 10 - General syntax for invoking the “convert" operation

- <inputType>: the input file format (i.e., json or xml)
- <inputPath>: the file-system path to the input file
- <outputType>: the desired output file format (i.e., json or xml)
- <outputPath>: the file-system path where the converted file will be written

An example invocation of the “convert” operation is shown in Figure 11.

```
SRIKernel convert buildingdata xml building.xml json
↪ building.json
```

Figure 11 - Example usage of the "convert" operation, converting building data from XML to JSON

- **Operation “compare”**

The “compare” operation compares two input files of the same data class and highlights their differences. It supports both building data and SRI score data, allowing comparisons between different input formats (JSON or XML). The general command-line syntax is shown in Figure 12.

```
SRIKernel compare <inputClass> <inputType1> <inputPath1>
↪  <inputType2> <inputPath2>
```

Figure 12 - General syntax for invoking the "compare" operation

The command parameters are defined as follows:

- <inputClass>: the class of data to be compared (i.e., buildingdata or scoresdata)
- <inputType1>: the format of the first input file (i.e., json or xml)
- <inputPath1>: the file-system path to the first input file
- <inputType2>: the format of the second input file (i.e., json or xml)
- <inputPath2>: the file-system path to the second input file

An example invocation of the “compare” operation is shown in Figure 13.

```
SRIKernel compare scoresdata json a.json xml b.xml
```

Figure 13 - Example usage of the "compare" operation, comparing two SRI score files in different formats

4. Data Structure and Requirements

4.1. Input Data Specifications (XML, JSON)

The “BuildingData” is the input data model. It consists of two mandatory top-level elements which are Building and Assessment. Both must be present for the calculator to operate.

First of all, for each method to calculate (Method A and Method B) different set of input data list is required. The data list required from the buildings for the calculation methods are listed down below in Figure 14 and 15.

Service Code	Service Name
H1a	Heat emission control
H1c	Storage and shifting of thermal energy
H2a	Heat generator control (all except heat pumps)
H2b	Heat generator control (for heat pumps)
H3	Report information regarding heating system performance
DHW1a	Control of DHW storage charging (with direct electric heating or integrated electric heat pump)
DHW1b	Control of DHW storage charging
DHW3	Report information regarding domestic hot water performance
C1a	Cooling emission control
C2a	Generator control for cooling
C3	Report information regarding cooling system performance
C4	Flexibility and grid interaction
V1a	Supply air flow control at the room level
V6	Reporting information regarding IAQ
L1a	Occupancy control for indoor lighting
DE1	Window solar shading control
DE4	Reporting information regarding performance of dynamic building envelope systems
E2	Reporting information regarding local electricity generation
E3	Storage of (locally generated) electricity
E11	Reporting information regarding energy storage
E12	Reporting information regarding electricity consumption
EV15	EV Charging Capacity
EV16	EV Charging Grid balancing
EV17	EV charging information and connectivity
MC13	Central reporting of TBS performance and energy use
MC25	Smart Grid Integration
MC30	Single platform for automated control + coordination of TBS with optimization based on occupancy, weather, and grid signals

Figure 14 - Service Codes and names for Method A

Service Code	Service Name
H1a	Heat emission control
H1b	Emission control for TABS (heating mode)
H1c	Control of distribution fluid temperature (air or water)
H1d	Control of distribution pumps in networks
H1f	Thermal Energy Storage (TES) for building heating (excluding TABS)
H2a	Heat generator control (all except heat pumps)
H2b	Heat generator control (for heat pumps)
H2d	Sequencing in case of different heat generators
H3	Report information regarding heating system performance
H4	Flexibility and grid interaction
DHW1a	Control of DHW storage charging (electric heating or integrated heat pump)
DHW1b	Control of DHW storage charging
DHW1d	Control of DHW storage charging (solar collector + supplementary heat)
DHW2b	Sequencing in case of different DHW generators
DHW3	Report information regarding domestic hot water performance
C1a	Cooling emission control
C1b	Emission control for TABS (cooling mode)
C1c	Control of chilled water temperature (supply/return)
C1d	Control of distribution pumps in cooling networks
C1f	Interlock: avoid simultaneous heating/cooling in the same room
C1g	Control of Thermal Energy Storage (TES) operation
C2a	Generator control for cooling
C2b	Sequencing of cooling generators
C3	Report information regarding cooling system performance

Service Code	Service Name
C4	Flexibility and grid interaction
V1a	Supply air flow control at room level
V1c	Air flow or pressure control at air handler level
V2c	Heat recovery control: prevention of overheating
V2d	Supply air temperature control at AHU
V3	Free cooling with mechanical ventilation
V6	Reporting information regarding IAQ
L1a	Occupancy control for indoor lighting
L2	Control of lighting power based on daylight
DE1	Window solar shading control
DE2	Window open/closed control (HVAC-integrated)
DE4	Reporting information on dynamic building envelope performance
E2	Reporting information on local electricity generation
E3	Storage of locally generated electricity
E4	Optimizing self-consumption of local generation
E5	Control of CHP plant
E8	Support of microgrid operation modes
E11	Reporting information regarding energy storage
E12	Reporting information regarding electricity consumption
EV15	EV Charging Capacity
EV16	EV Charging Grid balancing
EV17	EV charging information and connectivity
MC3	Runtime management of HVAC systems
MC4	Fault detection and diagnostics
MC9	Occupancy detection: connected services
MC13	Central reporting of TBS performance and energy use
MC25	Smart Grid Integration
MC28	Reporting DSM performance and operation
MC29	Override of DSM control
MC30	Single integrated control & optimization platform

Figure 15 - Service Codes and Method Names for Method B

- **JSON Structure**

Figure 16 illustrates the JSON structure of a “BuildingData” input file used by SRI Kernel. Next, we describe the implied constraints and valid value ranges of the individual fields.

```
{
    "Building": {
        "ResidentialStatus": "ResidentialStatusEnum",
        "Country": "CountryEnum"
    },
    "Assessment": {
        "Method": "MethodEnum",
        "Domains": [
            {
                "Name": "DomainEnum",
                "Presence": 0 | 1 | 2
            },
            ...
        ],
        "Services": [
            {
                "Name": "MethodAServiceEnum" |
                        "MethodBServiceEnum",
                "Triage": true | false,
                "Functionalities": [
                    {
                        "Level": -1 | 0 | 1 | 2 | 3 | 4,
```

Figure 16 - Indicative JSON structure of a "BuildingData" input file used by the SRI Kernel

The whole data structure is one JSON object enclosed in brackets (corresponding to the “BuildingData” object). That object must contain one “Building” object and one “Assessment” object.

Regarding the “Building” object:

- It must contain both “ResidentialStatus” and “Country”.
- “ResidentialStatus” takes value from the “ResidentialStatusEnum” enumeration.
- “Country” takes value from the “CountryEnum” enumeration.

It must contain all of the “Method”, “Domains”, “Services”, and “HasDefaultWeightings” fields. If “HasDefaultWeightings” is false, then “ImpactWeightings”, and “DomainWeightings” may not appear (if they appear, they will be ignored).

- “Method” takes value from “MethodEnum” enumeration.

Regarding the “Domains” collection:

- Every domain defined in the “DomainEnum” enumeration must appear exactly once in the collection of “Domains”.
- “Presence” must be one of 0 (absent), 1 (present), 2 (absent but mandatory).

Regarding the “Services” collection:

- Every service defined in the “MethodAServiceEnum” or “MethodBServiceEnum” enumeration (depending on what method you have declared in the “Method” field) must appear exactly once in the collection of “Services”.
- “Triage” takes a boolean value.
- Functionality levels must be valid for that service according to the SRI specification. Not all functionality levels apply to every service.
- Functionality level-1 corresponds to not applicable service. If that is the case, then the compliance must be 100% for that level, and no other functionality level can appear.
- Compliance values must sum to 100% per service.
- Under a specific service, a functionality level can appear at most one time.
- Services associated with domains of presence 0 or 2 are forbidden.

- HasDefaultWeightings" takes a boolean value.

Regarding the "ImpactWeightings" collection:

- There must be exactly one weighting per impact.
- "Impact" takes value from the "ImpactEnum" enumeration.
- Values must be between 0 and 1.
- Weightings must sum to 1.

Regarding the "DomainWeightings" collection:

- There must be exactly one weighting per (impact, domain) pair.
- "Impact" takes value from the "ImpactEnum" enumeration.
- "Domain" takes value from the "DomainEnum" enumeration.
- Values must be between 0 and 1.
- For each impact, the associated domain weightings must sum to 1.

XML Structure

Figure 17 illustrates the XML structure of a "BuildingData" input file used by SRI Kernel. Next, we describe the implied constraints and valid value ranges of the individual fields

Regarding the "Assessment" object:

The whole data structure is one XML element of type "BuildingData". That element must contain one "Building" element and one "Assessment" element.

```
<BuildingData>
    <Building>
        <ResidentialStatus>
            ResidentialStatusEnum
        </ResidentialStatus>
        <Country>
            CountryEnum
        </Country>
    </Building>
    <Assessment>
        <Method>MethodEnum</Method>
        <Domains>
            <Domain>
                <Name>DomainEnum</Name>
                <Presence>0 | 1 | 2</Presence>
            </Domain>
            ...
        </Domains>
        <Services>
            <MethodAService>
                <Name>
```

```
                MethodAServiceEnum
            </Name>
            <Triage>
                true | false
            </Triage>
            <Functionalities>
                <Functionality>
                    <Level>
                        -1 | 0 | 1 | 2 | 3 | 4
                    </Level>
                    <Compliance>
                        number
                    </Compliance>
                </Functionality>
            </Functionalities>
        </MethodAService>
        OR
        <MethodBService>
            <Name>
                MethodBServiceEnum
            </Name>
            <Triage>
                true | false
            </Triage>
            <Functionalities>
                <Functionality>
                    <Level>
                        -1 | 0 | 1 | 2 | 3 | 4
                    </Level>
                    <Compliance>
                        number
                    </Compliance>
                </Functionality>
            </Functionalities>
        </MethodBService>
        ...
    </Services>
    <HasDefaultWeightings>
        true | false
    </HasDefaultWeightings>
```

```
    <ImpactWeightings>
        <ImpactWeighting>
            <Impact>ImpactEnum</Impact>
            <Weighting>number</Weighting>
        </ImpactWeighting>
        ...
    </ImpactWeightings>
    <DomainWeightings>
        <DomainWeighting>
            <Impact>ImpactEnum</Impact>
            <Domain>DomainEnum</Domain>
            <Weighting>number</Weighting>
        </DomainWeighting>
        ...
    </DomainWeightings>
</Assessment>
</BuildingData>
```

Figure 17 - XML structure of a "BuildingData" input file used by SRI Kernel

Regarding the "Building" element:

- It must contain both "ResidentialStatus" and "Country" elements.
- "ResidentialStatus" takes value from the "ResidentialStatusEnum" enumeration.
- "Country" takes value from the "CountryEnum" enumeration.

Regarding the "Assessment" object:

- It must contain all of the "Method", "Domains", "Services", and "HasDefaultWeightings" elements. If "HasDefaultWeightings" is false, then "ImpactWeightings", and "DomainWeightings" may not appear (if they appear, they will be ignored).
- "Method" takes value from "MethodEnum" enumeration.

Regarding the "Domains" element:

- Every domain defined in the "DomainEnum" enumeration must appear exactly once in the "Domains" element.
- "Presence" must be one of 0 (absent), 1 (present), 2 (absent but mandatory).

Regarding the "Services" element:

- Every service defined in the "MethodAServiceEnum" or "MethodBServiceEnum" enumeration (depending on what method you have declared in the "Method" field) must appear exactly once in the "Services" element.
- "Triage" takes a boolean value.
- Functionality levels must be valid for that service according to the SRI specification. Not all functionality levels apply to every service.
- Functionality level-1 corresponds to not applicable service. If that is the case, then the compliance must be 100% for that level, and no other functionality level can appear.
- Compliance values must sum to 100% per service.
- Under a specific service, a functionality level can appear at most one time.
- Services associated with domains of presence 0 or 2 are forbidden.
- "HasDefaultWeightings" takes a boolean value.

Regarding the "ImpactWeightings" element:

- There must be exactly one weighting per impact.
- "Impact" takes value from the "ImpactEnum" enumeration.
- Weighting values must be between 0 and 1.
- Weighting values must sum to 1.

Regarding the "DomainWeightings" element:

- There must be exactly one weighting per (impact, domain) pair.
- "Impact" takes value from the "ImpactEnum" enumeration.
- "Domain" takes value from the "DomainEnum" enumeration.
- Weighting values must be between 0 and 1.
- For each impact, the associated domain weightings must sum to 1.

4.2. Output Data Specifications (XML)

- **ScoresData Structure**

The "ScoresData" is the output data model of the SRI calculations. It consists of five structured groups:

- "TotalScore"
- "AggregatedScores"
- "ImpactScores"
- "DomainScores"
- "DetailedScores"

- **JSON Structure**

Figure 18 illustrates the JSON structure of a “ScoresData” input file used by SRI Kernel. Next, we describe the implied constraints and valid value ranges of the individual fields.

Figure 18 - Indicative JSON structure of a "ScoresData" input file used by the SRI Kernel

The whole data structure is one JSON object enclosed in brackets (corresponding to the “ScoresData” object).

Regarding that object:

- It must contain all of the“TotalScore”,“AggregatedScores”,“ImpactScores”, “DomainScores”, and “DetailedScores” fields.
- “TotalScore” takes a value between 0 and 1.

Regarding the “AggregatedScores” object:

- It must contain all of the “AggregatedScoreOne”, “AggregatedScoreTwo”, and “AggregatedScoreThree” fields. Each one takes a value between 0 and 1.

Regarding the “ImpactScores” collection:

- It must contain exactly one object for each impact category defined in the “ImpactEnum” enumeration. Each such object, apart from the impact name, will contain a value between 0 and 1 corresponding to the score of that impact category.

Regarding the “DomainScores” collection:

- It must contain exactly one object for each impact category defined in the “DomainEnum” enumeration. Each such object, apart from the domain name, will contain a value between 0 and 1 corresponding to the score of that domain category.

Regarding the “DetailedScores” collection:

- It must contain exactly one object for each (impact,domain) pair defined in the “ImpactEnum” and “DomainEnum” enumerations, respectively. Each such object, apart from the impact and domain names, will contain a value between 0 and 1 corresponding to the score of that (impact,domain) pair.

- **XML Structure**

Figure 19 illustrates the XML structure of a “ScoresData” input file used by SRIKernel. Next, we describe the implied constraints and valid value ranges of the individual fields.

```
<SRIScores>
    <TotalScore>number</TotalScore>
    <AggregatedScores>
        <AggregatedScoreOne>number</AggregatedScoreOne>
        <AggregatedScoreTwo>number</AggregatedScoreTwo>
        <AggregatedScoreThree>number</AggregatedScoreThree>
    </AggregatedScores>
    <ImpactScores>
        <ImpactScore>
            <Impact>ImpactEnum</Impact>
            <Value>number</Value>
        </ImpactScore>
        ...
    </ImpactScores>
    <DomainScores>
        <DomainScore>
            <Domain>DomainEnum</Domain>
            <Value>number</Value>
        </DomainScore>
        ...
    </DomainScores>
    <DetailedScores>
        <DetailedScore>
            <Impact>ImpactEnum</Impact>
            <Domain>DomainEnum</Domain>
            <Value>number</Value>
        </DetailedScore>
        ...
    </DetailedScores>
</SRIScores>
```

Figure 19 - Indicative XML Structure of a "ScoresData" input file used by SRI Kernel

The whole data structure is one XML element of type “ScoresData”. Regarding that element:

- It must contain all of the “TotalScore”, “AggregatedScores”, “ImpactScores”, “DomainScores”, and “DetailedScores” elements.
- “TotalScore” takes a value between 0 and 1.

Regarding the “AggregatedScores” element:

- It must contain all of the “AggregatedScoreOne”, “AggregatedScoreTwo”, and “AggregatedScoreThree” elements. Each one takes a value between 0 and 1.

Regarding the “ImpactScores” element:

- It must contain exactly one element for each impact category defined in the “ImpactEnum” enumeration. Each such element, apart from the impact name, will contain a value between 0 and 1 corresponding to the score of that impact category.

Regarding the “DomainScores” element:

- It must contain exactly one element for each impact category defined in the “DomainEnum” enumeration. Each such element, apart from the domain name, will contain a value between 0 and 1 corresponding to the score of that domain category.

Regarding the “DetailedScores” element:

- It must contain exactly one element for each (impact,domain) pair defined in the “ImpactEnum” and “DomainEnum” enumerations, respectively. Each such element, apart from the impact and domain names, will contain a value between 0 and 1 corresponding to the score of that (impact,domain) pair.

5. Implementation Challenges & Mitigation Strategies

5.1. Technical Challenges

The technical integration of the Smart Readiness Indicator (SRI) within the EU Kernel EPC Engine introduces a set of challenges related to system heterogeneity, software interoperability, execution scalability, and methodological enforcement. These challenges arise from the need to operationalise a complex, multi-criteria assessment methodology within a federated and digitally diverse EPC ecosystem.

Across Member States, EPC tools differ significantly in terms of software architecture, programming languages, data models, and deployment environments. Directly embedding SRI logic into each national EPC tool would lead to fragmented implementations, increased maintenance effort, and a high risk of methodological inconsistencies. Additionally, EPC-related data are generated and stored in different formats across tools and platforms. Without strict data contracts, the exchange of information between EPC engines, assessment tools, and the SRI calculation component would be error-prone and difficult to automate.

To address this challenge, the SRI calculation logic is encapsulated within a standalone, cross-platform SRI Kernel Engine. The Kernel is distributed as self-contained executables for major operating systems and can be invoked externally by the EU Kernel EPC Engine. This decoupling allows national EPC tools to interact with a single authoritative SRI implementation without requiring internal modification of their codebases. Moreover, the SRI Kernel enforces the use of standardised, machine-readable input and output structures based on XML and JSON. Dedicated validation and conversion functions ensure that data exchanged between systems adhere to predefined schemas. This approach reduces integration friction and enables the EU Kernel EPC Engine to orchestrate SRI calculations reliably across heterogeneous digital environments.

To improve the SRI Kernel Engine's robustness, compatibility, and long-term maintainability, more technical improvements can be anticipated in addition to the fundamental architectural changes mentioned above. In order to support future methodology evolution without interfering with current integrations, these include formalizing the current input and output data models through the publication of explicit JSON Schema and XML Schema Definition (XSD) files along with precisely defined backward-compatibility rules. In order to improve automation reliability and system integration transparency, deterministic error handling tools can be developed. These include defined exit codes, structured error objects, and a clear distinction between blocking validation errors and non-blocking warnings. A set of reference test vectors, consisting of at least two complete and validated sample inputs for both Method A and Method B with matching expected outputs, will be maintained in order to facilitate regression testing and cross-release consistency. Additionally, by either expanding the output to include explicit difference reporting or by aligning documentation with its boolean equivalency logic, the functional behavior of the comparison operation will be explained to guarantee consistency between its written purpose and its actual output. In order to facilitate well-informed deployment decisions in large-scale EPC registries and national databases, performance indicators such as indicative runtime per calculation, memory footprint, and appropriateness for batch and registry-scale processing will be provided.

Secondly, the SRI methodology includes numerous constraints, such as mandatory domain declarations, method-specific service catalogues (Method A and Method B), functionality-level bounds, partial compliance rules, and strict weighting normalisation. Implementing these constraints inconsistently across tools would undermine the credibility and comparability of SRI results. All methodological rules are enforced centrally within the SRI Kernel. Before any calculation is performed, the kernel applies automated validation checks to ensure full compliance with the official SRI framework. This guarantees that SRI scores integrated into EPC outputs are methodologically sound and comparable, regardless of the originating EPC tool or data source.

5.2. Data Availability and Standardisation

The SRI methodology requires detailed and structured information on building systems, smart-ready services, functionality levels, and applicability conditions. In practice, ensuring the consistent availability and harmonisation of such data across diverse building contexts and Member States represents a significant challenge.

Data required for SRI assessments originate from multiple and heterogeneous sources, including EPC software tools, on-site audits, technical documentation, digital building models, and assisted data-collection channels such as call centres. These sources vary in their level of detail, terminology, and reliability, leading to potential gaps or inconsistencies when assembling SRI input dataset. Within OpenBEP4EU, data heterogeneity is addressed upstream of the SRI Kernel through harmonisation processes implemented in the

Secondly, the SRI methodology requires explicit identification of applicable services and their functionality levels across all relevant domains. In real-world assessments, certain services may be undocumented, partially known, or difficult to assess due to limited access to technical information, especially in existing buildings. Additionally, due to limited financial resources and untrained personnel, it can be difficult for obtaining the required or the desired data for the SRI Kernel Engine parameters to compute. Therefore, the SRI Kernel enforces explicit declaration of service applicability and functionality. Where services are not applicable or domains are absent, this must be explicitly stated in the input data, in line with the official SRI framework. This approach increases transparency and avoids artificially inflated or misleading SRI scores caused by missing information.

Additionally, differences in terminology, system classification, and national conventions can hinder the consistent interpretation of building services and smart functionalities. Without standardisation, equivalent systems might be described differently across datasets, reducing comparability. The SRI Kernel relies on predefined service catalogues and domain definitions derived from the official SRI calculation framework. Input data must reference these standardised identifiers and structures, ensuring consistent classification of services and functionalities across assessments. This standardisation supports comparability of SRI results across buildings, regions, and Member States.

5.3. Cross-Country Harmonisation

In order to translate this harmonized philosophy into regular practice, the OpenBEP4EU SRI Kernel is essential. Regardless of the country environment in which an assessment is conducted, the kernel guarantees that the same rules, limitations, and weighting principles are implemented by implementing the SRI calculation logic as a single, authoritative computational engine. This centralized execution mechanism facilitates the production of similar outcomes across Member States and lowers the possibility of differing interpretations of the technique. The SRI framework's explicit consideration of contextual and climatic variations further supports cross-country harmonization. The methodology's default weighting variables take climatic zone and building type changes into consideration, enabling SRI scores to represent regional and national specificities without sacrificing general comparability. System-level harmonization is strengthened by the SRI Kernel's interaction with the EU Kernel EPC Engine. The EU Kernel guarantees that SRI assessments are integrated into EPC workflows using standardized data structures, execution sequences, and result formats by acting as a common orchestration layer. This allows national EPC systems to maintain control over local data collecting and display procedures while interacting with a common computational backbone.

The interpretation and repurposing of SRI results is another aspect of harmonization. Overall scores, impact-level scores, domain-level scores, and detailed matrices are all included in the structured output produced by the SRI Kernel and are all presented in a consistent manner. This uniformity facilitates analysis and policy evaluation at the EU level by supporting cross-national aggregation, benchmarking, and monitoring of smart

readiness across building stocks. The framework permits controlled configuration alternatives, such as predetermined assessment techniques or weighting selections in line with country implementation strategies, while maintaining harmonization of the fundamental SRI methodology and calculation logic. Transparency and traceability are maintained by the kernel's clear declaration and processing of these choices.

While the EU Kernel offers a harmonized execution environment for approaches that must be performed consistently at the European level, existing EPC tools continue to function as the main interfaces for data collecting and user interaction. The addition of SRI capability is made possible by this layered architecture, which eliminates the need to completely redesign national EPC tools. EPC tools accept SRI results in an organized manner that can be integrated into EPC outputs, reports, or registries, and they supply the required building and system information either directly or through intermediate harmonization stages. This has a number of useful integration benefits. First, by enabling both more sophisticated digital interfaces and file-based data interchange, it maintains compatibility with legacy EPC systems. Second, it guarantees that SRI computations are carried out using a single, authoritative implementation of the approach, regardless of the internal logic of different EPC instruments. From the view of end users, including building experts and EPC assessors, the integration strategy seeks to reduce extra complexity. Instead of needing assessors to directly manage the underlying calculation rules, the EU Kernel and SRI Kernel handle methodological logic and validation centrally, enabling EPC solutions to expose SRI-related capabilities through guided workflows or automated procedures

6. Conclusion & Next Steps

6.1. Conclusion

By converting the established spreadsheet-based technique into a structured, automated, and reusable digital service, the engine constitutes a comprehensive and functional software application of the official European SRI calculation framework. The whole SRI calculation logic, including data validation, multi-level aggregation, weighing, normalization, service and functionality-level evaluation, and result creation, is contained in the SRI Kernel Engine. It upholds all methodological restrictions established at the EU level and supports both the official SRI assessment methodologies, Method A (simple) and Method B (detailed). The kernel guarantees consistent and repeatable SRI results across various tools, workflows, and national settings by centralizing these principles inside a single execution engine.

Technically speaking, the engine is a stand-alone, cross-platform solution that the EU Kernel EPC Engine can easily orchestrate. Its compatibility with current EPC tools, digital building platforms, and data repositories is made possible by its reliance on standardized, machine-readable input and output architectures. Advanced use cases like longitudinal analysis and quality control are supported by the kernel's capacity to validate, transform, and compare datasets, which further improves data quality assurance.

The SRI Kernel Engine is a crucial component in integrating smart readiness into next-generation Energy Performance Certificates at the system level. The engine makes it possible to consistently include smart readiness indications with conventional energy performance measurements by incorporating SRI computations into digital EPC procedures. This enhances EPCs' analytical value and encourages their application as tools for monitoring, policy implementation, and refurbishment planning.

The built engine offers a basis for connecting technical building assessments with more general market and financing applications, going beyond regulatory compliance. Through the EU Kernel, external stakeholders can access the organized and comparative SRI outputs produced by the kernel to design finance structures that acknowledge the value of smart and digitally integrated buildings.

6.2. Next Steps

The revised EPBD (EU) 2024/1275 introduces requirements for non-residential buildings to be equipped with BACS above defined capacity thresholds and specifies minimum BACS capabilities (continuous monitoring/analysis, benchmarking and loss detection, interoperability, and IEQ monitoring by 29 May 2026). A practical verification approach is operationalised in the *eu.bac 2024 EPBD BACS Compliance Verification Checklist for Non-Residential Buildings*, which structures compliance into (i) eligibility by capacity, (ii) minimum BACS coverage, and (iii) capability checks organised by Article 13(10)(a)–(d) [3] [5].

A logical next step for the EU Kernel EPC architecture is to add an optional **BACS compliance extension** that reuses the existing EU Kernel data model and the validated SRI Kernel input dataset to generate a **semi-automated compliance pre-check**. While many SRI-related declarations align conceptually with BACS capability requirements (e.g., scheduling, central setpoint management, monitoring/trending, fault detection and diagnostics), some checklist items require operational evidence (e.g., hourly metering integration, benchmarking reports, alarm/activity logs, IEQ trend reports) [5].

Concretely, the extension should: (1) implement Step 1 and Step 2 eligibility/coverage rules (including the checklist's 80% coverage criteria), (2) provide a documented crosswalk between SRI services/functionality levels and checklist IDs (A1–D1), (3) output a structured compliance report (pass/fail per checklist ID),

explicitly flagging "data gaps / evidence required" where compliance cannot be concluded from declared inputs alone, and (4) enable export of a compliance dossier to support inspections and reporting workflows [5].

A. Eligibility & Coverage (Step 1 & Step 2) [5]

- Heating / cooling capacity thresholds (290 kW / 70 kW)
- ≥80% coverage of heating, cooling, ventilation outputs
- ≥80% coverage of continuously occupied floor area
- Provide a documented crosswalk between SRI services/functionality levels and checklist IDs (A1–D1)

B. Compliances (Step 3 & Step 4) [5]

- Continuous monitoring, logging, and analysis
- Structured compliance report (pass/fail per checklist ID), **explicitly flagging "data gaps / evidence required"** where compliance cannot be concluded from declared inputs alone
- Cross-system optimisation
- Interoperability across devices and manufacturers
- Enabling the export of a compliance dossier to support inspections and reporting workflows

This evolution would position the EU Kernel as a reusable digital backbone supporting both performance-based smart readiness assessments and EPBD compliance verification, without duplicating data collection processes.

7. References

1. SRI Commission Delegated Regulation (EU) 2020/2155 Europa.eu. [cited 2026 Jan 27]. Available from: https://eur-lex.europa.eu/eli/reg_del/2020/2155/oj/eng

2. SRI Commission Implementing Regulation (EU) 2020/2156 Europa.eu. [cited 2026 Jan 27]. Available from: https://eur-lex.europa.eu/eli/reg_impl/2020/2156/oj/eng

3. Directive - EU - 2024/1275 - EN - EUR-Lex [Internet]. Europa.eu. [cited 2026 Jan 23]. Available from: https://eur-lex.europa.eu/eli/dir/2024/1275/oj/eng

4. SRI implementation tools [Internet]. Energy. [cited 2026 Jan 27]. Available from: https://energy.ec.europa.eu/topics/energy-efficiency/energy-performance-buildings/smart-readiness-indicator/implementation-tools_en

5. Eubac.org. [cited 2026 Jan 23]. Available from: https://eubac.org/wp-content/uploads/2024/06/2024_EPBD_BACS-COMPLIANCE-VERIFICATION-CHECKLIST_PACKAGE_non-resident_EN.pdf