International Standard

ISO/IEC 21031

Information technology — Software Carbon Intensity (SCI) specification

Technologies de l'information — Spécification relative à l'intensité carbone logicielle

First edition
2024-03

Reference number
ISO/IEC 21031:2024(en)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by the Linux Foundation (as Software Carbon Intensity (SCI) Specification, v.1.0) and drafted in accordance with its editorial rules. It was adopted, under the JTC 1 PAS procedure, by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

“If you can't measure it, you can't improve it.” – Peter Drucker

Software systems cause emissions through the hardware that they operate on, both through the energy that the physical hardware consumes, and the emissions associated with manufacturing the hardware. This specification defines a methodology for calculating the rate of carbon emissions for a software system. The purpose is to help users and developers make informed choices about which tools, approaches, architectures, and services they use in the future. It is a score rather than a total; lower numbers are better than higher numbers, and reaching 0 is impossible. This specification is focused on helping users and developers understand how to improve software to reduce or avoid the creation of emissions.

Reducing an SCI score is only possible through the elimination of emissions. That can be achieved by modifying a software system to use less physical hardware, less energy, or consume lower-carbon energy sources. Neutralization or avoidance offsets do not reduce an SCI score (Clause 12). This makes the SCI an ideal strategy that organizations can adopt to meet climate targets focused on eliminating emissions, such as those specified by[1].

The SCI is for everyone. It is possible to calculate an SCI score for any software application, from a large, distributed cloud system to a small monolithic open source library, any on-premise application, or even a serverless function. The environment the product or service is running in can also vary; from personal computers, private data centers or a hyperscale cloud.

Software practitioners have a significant role to play in collectively reducing the SCI score during the design, development, and delivery of software applications. The following list provides some strategies that can be used to do this across different software roles:

— For a software programmer, this implies writing energy efficient code.

— For an AI/ML developer, it implies model optimization, using pre-trained models or leveraging optimized hardware for training.

— For a database engineer, this comprises choices like schema design, choice of storage, and query optimizations.

— For a DevOps practitioner, this requires creating a carbon-aware pipeline and considering when to schedule builds and leverage clean energy.

— For QA engineers, it involves creating energy efficient test automation and performance testing scripts across browsers and devices.

— For an architect, this implies choices like serverless or event driven architectures, infrastructure optimization, and design for carbon-aware systems.

The SCI encourages calculation using granular real-world data, which is challenging to obtain in some environments, particularly the public cloud. Access to the data needed for higher resolution calculations might not always be available.

Where this is the case, users of this specification are strongly advised to request such data from their suppliers (be they hardware, hosting, or other).

In situations where there is a lack of access, capability, or rights to the necessary real-world data, the SCI allows for data generated through modelling, using best estimates instead.

Information technology — Software Carbon Intensity (SCI) specification

1 Scope

This specification describes a methodology for calculating the rate of carbon emissions for a software system; that is, its SCI score. The purpose of this score is to increase awareness and transparency of an application's sustainability credentials. The score will help software practitioners make better, evidence-based decisions during system design, development, and deployment, that will ultimately minimize carbon emissions. A reliable, consistent, fair and comparable measure allows targets to be defined during development and progress to be tracked.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

2.1
action
explicit outcome taken, or change avoided, depending on the quantifiable emissions measured by this specification

Note 1 to entry: Note to entry: Actions generally relate to using less electricity, using electricity more intelligently, or using less hardware.

2.2
carbon-aware
attribute of software or hardware that adjusts its behavior (consumption of inputs, processing, or production of outputs) in response to the carbon intensity of the energy it consumesThe following abbreviations are used throughout this specification:

— E – Energy consumed by a software system

— I – Location-based marginal carbon intensity

— M – Embodied emissions of the hardware needed to operate a software system

— O – Operational emissions based on the emissions caused by energy consumption

— R – Functional unit

4 Software sustainability actions

All actions that serve to reduce the carbon emissions of a piece of software fit into one of the following categories:

— **Energy Efficiency**: Actions taken to make software use less electricity to perform the same function.

— **Hardware Efficiency**: Actions taken to make software use fewer physical resources to perform the same function.

— **Carbon Awareness**: Actions taken to time- or region-shift software computation to take advantage of cleaner, more renewable or lower carbon sources of electricity.

It is the intent of this specification to encourage more of these actions to be taken during the design, development, and maintenance of software applications.

5 Procedure

The steps required to calculate and report an SCI score are:

1. **Bound**: Decide on the software boundary (Clause 6); i.e., the components of a software system to include.
2. **Scale**: As the SCI is a rate (carbon emissions per one functional unit [Clause 7]), pick the functional unit which best describes how the application scales.
3. **Define**: For each software component listed in the software boundary, decide on the quantification method (Clause 8); real- world measurements, based on telemetry, or lab-based measurements, based on models.
4. **Quantify**: Calculate a rate for every software component. The SCI value of the whole application is the sum of the SCI values for every software component in the system.
5. **Report**: Disclose the SCI score, software boundary, and the calculation methodology.

6 Methodology summary

6.1 General

SCI is a rate; carbon emissions per one unit of R. The equation used to calculate the SCI value of a software system is:

SCI = C per R

Where:

1. The total amount of carbon C the software causes to be emitted.
2. All the elements in the SCI equation scale by the same functional unit of R (e.g., carbon emissions per additional user, API-call, or ML training run).

This can be expanded to:

SCI = (O + M) per R

6.2 Operational emissions

6.2.1 General

To calculate the operational emissions associated with software, multiply the electricity consumption of the hardware the software is running on by the regional, granular marginal emissions rate. The marginal emissions rate reflects the change in emissions associated with a change in demand.

To calculate the operational emissions O for a software application, use the following:

$$O = (E * I)$$

6.2.2 Energy

This is the energy consumed by a software system for a functional unit of work. This could be applied to several taxonomies:

— Datacenter

— Individual machine (e.g., VM/Node)

— Individual service (e.g., API call or ML training run)

— Execution of code

Units: this shall be in kilowatt hours (kWh).

6.2.3 Location-based marginal carbon intensity

The carbon intensity of electricity is a measure of how much carbon (CO2eq) emissions are produced per kilowatt-hour (kWh) of electricity consumed. Because this specification uses a consequential approach, marginal emissions rates shall be used for electricity consumption.

Location-based marginal emissions factors measure the grid carbon intensity of a grid region. If the electricity consumption is connected to a grid, the marginal emissions rate of that grid shall be used, which excludes any market-based measures (11.2). If the electricity consumption is not connected to a larger regional grid, an appropriate emissions factor for that system shall be used. From a developer perspective, only the location-based info is important in terms of the impact on eliminating carbon emissions. This excludes market-based measures and is distinct from 100% renewable energy claims.

The only figure that matters when trying to optimize the scheduling of a computation in real-time is the marginal emissions intensity. This is the emissions intensity of the marginal power plant which will need to be turned up if a computation is scheduled (e.g., increase electricity demand from the grid) at that moment.

Units: this shall be in grams of carbon per kilowatt hours (gCO2eq/kWh).

6.3 Embodied emissions

Embodied carbon (otherwise referred to as “embedded carbon”) is the amount of carbon emitted during the creation and disposal of a hardware device.

When software runs on a device, a fraction of the total embodied emissions of the device is allocated to the software.

This is the value of M that needs to be calculated in the SCI equation.

This fraction consists of both a time- and resource-share. The length of time that the software runs on the device determines its time-share. The percentage of the device reserved just for that application during the time-share determines that application's resource-share.

To calculate the time-share, amortize the total embodied carbon over the expected life span of the device and then extrapolate based on the time reserved for the usage. For example, if the device's embodied carbon was 1000 kg with an expected lifespan of four years and it was reserved for use for one hour, the time-share embodied emissions would be 1000 * 1/(4*365*24) or around 28 g of the total.

To calculate resource-share, look at the share of total available resources reserved for use by the software. For instance, the percentage of total virtual CPUs reserved for the software is a good choice for the resource-share metric in the virtualized cloud space.

To calculate the share of M for a software application, use the equation:

M = TE * TS * RS

Where:

— TE = Total Embodied Emissions; the sum of Life Cycle Assessment (LCA) emissions for all hardware components.

— TS = Time-share; the share of the total lifespan of the hardware reserved for use by the software.

— RS = Resource-share; the share of the total available resources of the hardware reserved for use by the software.

The equation can be expanded further:

M = TE * (TiR/EL) * (RR/ToR)

Where:

— TiR = Time Reserved; the length of time the hardware is reserved for use by the software.

— EL = Expected Lifespan; the anticipated time that the equipment will be installed.

— RR = Resources Reserved; the number of resources reserved for use by the software.

— ToR = Total Resources; the total number of resources available.

An estimate of all the embodied emissions for the hardware used within the software boundary shall be included.

Simple models to estimate embodied emissions may be used; however, the most granular data possible and ideally emissions data from a device's LCA when calculating the embodied carbon should be used.

Since the purpose of the SCI is the elimination of emissions M shall not include any market-based measures (11.2).

Units: this shall be in grams of carbon (gCO2eq).

6.4 Functional unit conversion

An aggregate SCI score can be composed of multiple component SCI scores.

Then, as long as the functional unit of R is the same across all the component SCI scores, these can be summed to calculate the aggregate SCI. To sum multiple component SCI scores into one aggregate score, the functional unit R shall be the same across all components.

If the functional unit of a software component is not the same as the aggregate functional unit, then the component SCI score needs to be converted to match that of the aggregate SCI functional unit. Details of any unit conversion factors used in calculating the SCI score shall be disclosed.

7 Software boundary

The first step in generating an SCI score is deciding what the boundaries of the software system are; i.e., what software components to include or exclude in the calculation of the SCI score.

The calculation of SCI shall include all supporting infrastructure and systems that significantly contribute to the software's operation.

Supporting infrastructure and systems may include:

— compute resources

— storage

— networking equipment

— memory

— monitoring

— idle machines

— logging

— scanning

— build and deploy pipelines

— testing

— training ML models

— operations

— backup

— resources to support redundancy

— resources to support failover

— End user devices

— IoT devices

— Edge devices

If the boundary includes on-premise and/or cloud data center operations, E should take into account the efficiency of the data center, including cooling and other energy consumption necessary to operate a data center. The data center's energy efficiency is usually available as a PUE (Power Usage Effectiveness) value.

8 Functional unit

The second step in generating an SCI score is deciding which functional unit will be used to describe how the application scales. First, decide on the functional unit, using the choice of R. Then calculate how much C is emitted per unit of R.

For instance, if the application scales by number of users, then choose this as the functional unit.

A consistent choice of R across all the components in the software boundary shall be used.

A suggested list of functional units includes:

— API call/request

— Benchmark

— User

— Machine

— Minute/time unit

— Device

— Physical site

— Data volume

— Batch/Scheduled Job

— Transaction

— Database read/write

9 Quantification method

9.1 General

The third step in generating an SCI score is deciding the approach to take when quantifying the carbon emissions for each component in the software boundary.

The goal of the SCI is to quantify how much C (carbon) is emitted per one unit of R.

There are two main approaches to quantifying carbon emissions (C), measurement (8.2) via real-world data or calculation (8.3) via models.

Each component in the software boundary may use either measurement or calculation to quantify the carbon emissions.

It is strongly advised that suppliers (be they hardware, hosting, or other) be contacted regarding the data needed in the resolution required for quantifying the SCI score.

9.2 Measurement

Carbon emissions may be quantified by measuring the total real-world carbon emissions of the component (C) over a time period and dividing by the number of functional units (R) in the same time period to get C per R. For instance, data regarding the real-world usage of the application "in the wild" might be measured and then divided by the number of users serviced in the same time period to get C per user.

9.3 Calculation

What one unit of R looks like may be modelled and the total carbon (C) calculated for executing one functional unit of work (R) in a controlled lab environment. For instance, a benchmark application may be created that models a user interacting with **developer's** and then measure the C emitted per run of that benchmark. The result is still a C per user.

10 Comparing an SCI score to a baseline

When taking an action to reduce the carbon intensity of a piece of software, the intensity should be compared to a baseline. The baseline shall be calculated using an identical methodology to how the proposed SCI was calculated, except excluding the proposed action(s). The measurements, assumptions, models, functional units, etc. shall remain the same between the baseline and proposed SCI.