Functional Units for SCI in AI Systems

(Draft for Discussion – Green Software Foundation, AI Workgroup)
This document is a working draft to stimulate discussion and alignment within the GSF Standards – SCI for AI Workgroup. It is not a finalized specification.

Purpose

This draft outlines multiple **functional unit options** for various AI-related use cases and roles. It provides context and rationale for each, aiming to:

- Align SCI metrics with real-world AI system usage
- Enable meaningful and actionable carbon accounting
- Reflect the complexity of roles like API consumers, AI developers, and multi-agent systems
- Inform the evolution of the **Software Carbon Intensity (SCI)** standard to support AI-specific needs

Functional Units by Use Case

---

1. End Users (Individuals and Organizations)

End users interact with AI services to perform tasks. Their SCI interest lies in **per-use efficiency** and **organizational reporting (e.g., ESG, Scope 3)**.

Functional Unit	Definition	When to Use
Per query or API call	One prompt sent to an LLM	Default for API-based inference
Per functional output	One completed task (e.g., image, doc)	User-focused, multi-modal systems
Per session	Multiple interactions in one flow	For chatbots or workflows
Per user per month/year	Aggregated usage per person	Internal org dashboards
Per organization per quarter/year	Total org usage	ESG or Scope 3 reporting

Measurement Approach :

- Use emissions per API call as calculated in the **Evaluating.the.Inclusion.of.LLM.Training.Emissions.in.SCI.for.API.Consumers document)**

- SCI = `emissions (gCO₂e) / call`
- In the **extended SCI boundary**, also consider amortized training emissions per call (if available):

```
SCI = E_infer + (E_train × N_entity / N_total)
```

  - `E_train` (total emissions from training)
  - `N_total` (expected usage over model life)
  - `N_entity` (user-specific usage)

2. AI Researchers, Developers, and Engineers

These professionals create and optimize models. Their SCI interests include training efficiency, hardware use, and model architecture choices.

Functional Unit Options:

Functional Unit	Definition	When to Use
Per model trained	Full emissions of one training job	Benchmarking large-scale training
Per epoch	One full pass through training data	Hyperparameter tuning or iteration tracking
Per million parameters	Emissions normalized by model size	Comparing architectures
Per FLOP used	Per floating point operation	Fine-grained efficiency analysis
Per inference	Emissions per deployed model prediction	Deployment optimization
Per GPU-hour	Hardware time allocation	Infrastructure-level efficiency tracking

Measurement:

- For internal teams, direct monitoring tools can capture power usage and training time.
- Use embodied emissions and runtime energy to derive per-epoch or per-parameter values.
- Tracking through logs, job schedulers, and emissions APIs is essential for validation.

3. AI Companies (Model/Service Providers)

Companies offer AI services to other organizations and must account for both training and inference stages, as well as guide consumers on low-carbon options.

Functional Unit Options:

Functional Unit	Definition	When to Use
Per API request	One inference by a client	Common billing and benchmarking unit

Functional Unit	Definition	When to Use
Per customer per month/year	Aggregated by org	Enables client-facing ESG support
Per model lifecycle	Training + deployment	Lifecycle tracking and internal auditing
Per deployment instance	Per endpoint or container	Cloud-native, serverless applications
Per compute region	Regionally segmented emissions	Geolocation-based reporting or carbon-aware routing

Measurement Guidance:

- Use SCI boundaries defined for LLM API consumers in the discussion documents.
- Encourage use of **inference-only** SCI as default, and **amortized training SCI** where provider disclosure allows it.
- Provide emissions per model version and enable model selection transparency for clients.

4. Agentic AI Workflows (Multi-Model Use Cases)

Agentic workflows involve multiple AI components working together. They demand hybrid SCI strategies, where multiple models may be used for a single user task.

✅ **Functional Unit Options:**

Functional Unit	Definition	When to Use
Per composite task	One full end-to-end user task	Complex workflows or planning agents
Per agent interaction	One message or command between agents	For debugging and optimization
Per scenario execution	Full system execution (e.g., travel planning)	Scenario-based modeling

SCI Calculation Based on Agentic Use Case:

- Refer to Agentic-AI-Usecase.pdf in discussion documents for calculation and more details.
- Use model-specific SCI scores ($\texttt{gCO}_2\texttt{e / call}$) from actual inference data.
- Example: gpt-4o = 42.4478 gCO_2e/call vs gpt-4o-mini = 2.3785 gCO_2e/call.
- Combine inference + overhead emissions from orchestration tools (e.g., Agent SDK).
- Exclude tools if they are outside the system boundary (e.g., external API calls) or if no emissions measurement is provided by the third-party tool provider. This ensures SCI calculations remain grounded in measurable and attributable components only.
- If training emissions are included, apply amortization formula across all agents involved.

Measuring Emissions: Insights from Previous discussions

From the previous discussion documents:

1. **Inference-only SCI** is the default and most practical for API users, aligned with how managed services are handled.
2. **Amortization of training emissions** is possible if providers disclose:
   - `E_train` (total emissions from training)
   - `N_total` (expected usage over model life)
   - `N_entity` (user-specific usage)
3. **Extended SCI boundaries** should be optional and clearly labeled.
4. Standardized usage metrics (e.g., token count, prompt count) are needed for fair allocation.
5. Training emissions are **included by default** for model creators, fine-tuners, and self-hosted deployments.

Disclaimer

This is a working draft created to support discussions within the Green Software Foundation for SCI for AI Workgroup.
It is not an official or final standard. Functional units and measurement methods are evolving, especially for complex scenarios such as multi-agent systems and managed APIs.