## The Interest of SONNX in the Context of ED-324/ARP6983

The primary interest of SONNX is to transform the MLMD from a simple data-exchange file into a formal, verifiable, and traceable specification suitable for certifying safety-critical systems under ED-324/ARP6983.

The ED-324 standard places the MLMD in a critical role: it is the definitive blueprint that bridges the gap between the data science-oriented model design phase and the safety-oriented hardware/software implementation phase. For a certification argument to be successful, there must be indisputable proof that the final, implemented product is a faithful and complete representation of this blueprint.

Standard ONNX, while popular, fails to provide this level of assurance due to ambiguities, implicit behaviors, and insufficient specification. SONNX's interest lies in solving this exact problem by providing the necessary rigor. Its key benefits are:

* **Ensuring Unambiguous Specification:** SONNX replaces vague operator definitions with precise mathematical and formal specifications. This eliminates interpretation and ensures that a development team can implement the model exactly as intended by the designers, which is a fundamental prerequisite for any certification activity.
* **Enabling Verifiability:** The standard requires extensive verification to show the implementation preserves the model's properties (like performance and stability) on the target hardware. SONNX directly supports this by providing executable oracles, in the form of its formal specification and a reference C implementation, against which the final, optimized code can be rigorously tested.
* **Guaranteeing Traceability:** ED-324 mandates a clear, auditable trail from requirements to implementation. By prohibiting implicit behaviors like automatic shape inference and default values, SONNX ensures the MLMD is an explicit and complete specification, making the link between the model design and its implementation transparent and traceable.

### ED-324 Requirements Answered by SONNX as an MLMD Format

SONNX is designed to directly address the following requirements and objectives outlined in ED-324/ARP6983:

* **Requirement for a Complete and Unambiguous Model Description:**
  + ED-324 requires the MLMD to capture the model's full "analytical/algorithmic syntax and semantics".
  + **SONNX answers this** by providing a three-tiered specification for each operator: a clear informal document, a mathematically sound formal specification (WhyML), and a traceable reference implementation, leaving no room for ambiguity.
* **Requirement for Verifying the Final Implementation against the Design:**
  + ED-324 has numerous objectives for verifying that the final integrated MLC, running on the target hardware, behaves as specified (Section 6.3). For example, performance must be verified against the test dataset.
  + **SONNX answers this** by providing trusted oracles for comparison. The output of the final product can be directly compared against the output of the executable formal specification or the simple C implementation to prove conformance.
* **Requirement for Bi-Directional Traceability:**
  + A key objective in ED-324 is to ensure that the MLMD is "completely developed into the MLMIDs" (the item-level implementation descriptions).
  + SONNX answers this by making the model description fully explicit. By removing implicit features like shape inference and default values, the SONNX file becomes a "what you see is what you get" blueprint, creating a clean and auditable trace from the design artifact to the implementation artifact.
* **Requirement for Configuration Management and Reproducibility:**
  + As a supporting process, ED-324 requires rigorous configuration management to ensure that all development artifacts are identified, controlled, and retrievable (Table F1).
  + **SONNX answers this** by providing a stable, deterministic, and complete format. A SONNX file can be reliably archived and used to reproduce the exact same model behavior, which is essential for managing baselines, assessing changes, and maintaining the certified configuration over the product's lifecycle.

## 

## 

## 

## 

## 

## 

## 

## ANNEXES

## SONNX to serve as a certifiable version of the MLMD (Machine Learning Model Description)

Based on ED-324/ARP6983, the MLMD (Machine Learning Model Description) is the comprehensive technical specification that formally documents the complete design and final configuration of a trained ML model. It goes beyond a high-level summary to detail the model's exact analytical and algorithmic semantics, which includes its logical architecture, all hyperparameters that controlled the learning process, and the final trained parameters (e.g., neural network weights). This artifact serves as the definitive blueprint for the implementation phase, providing all the necessary information to compute the model's outputs from its inputs and ensuring that the model can be correctly and verifiably replicated in the final hardware or software product.

### Purpose of SONNX for the MLMD

The Machine Learning Model Description (MLMD) is a pivotal artifact in the development lifecycle of ML-based systems. According to the ARP6983/ED-324 standard, the MLMD is the final output of the model design phase (first V of the Wshaped process) and serves as the primary input specification for the implementation phase (second V). It must capture the full semantics of the trained model to be transformed into software and/or hardware items.

The standard explicitly requires demonstrating that:

* The MLMD completely and correctly represents the ML Model.
* The final implementation, described in the ML Model Item Description (MLMID), is fully traceable to the MLMD.

The purpose of using the SONNX profile to express this MLMD is to ensure that this specification is:

* **Unambiguous and Rigorous:** It must be perfectly clear and non-ambiguous, allowing an implementer to interpret the model exactly as the designer intended.
* **Verifiable:** It enables the demonstration that the model's semantics are preserved throughout the implementation phase, a key requirement for certification in domains like aeronautics.
* **Interoperable in a Critical Context:** While standard ONNX provides interoperability between ML tools (like PyTorch and TensorFlow), SONNX aims to extend this to the highly-regulated toolchains used for developing and certifying critical systems.

In essence, SONNX's purpose is to formalize the ONNX standard, making it a robust and trustworthy blueprint for creating certifiable ML items from a trained model.

## Challenges with Standard ONNX as an MLMD

While ONNX is a popular standard for exchanging ML models, its design for "general purpose AI" presents significant challenges from a safety and certification perspective. The SONNX working group has identified several challenges:

* **Insufficient Specification:** The documentation for many standard ONNX operators is too vague for rigorous implementation. For example, the Conv operator is simply described as consuming an input and a filter to compute an output, lacking the precise mathematical detail needed for verification.
* **Ambiguity Left to Implementers:** The standard explicitly allows for implementation-defined behaviors. For operator overloading, the documentation states that in case of conflicts, the behavior is "defined by the runtimes. Similarly, the exact execution order of operators is not mandated, which, while semantically flexible, reduces determinism.

#### Challenge 1: Ambiguity and Insufficient Specification

* **The Problem:** The certification process requires a complete and unambiguous specification to ensure the implemented model is identical to the designed one. However, the standard ONNX format has vague operator definitions and leaves some behaviors up to the interpretation of the runtime engine. This ambiguity is unacceptable for certification, where every detail must be precisely defined and verified.
* **SONNX Solution:** SONNX solves this by providing three layers of specification for each operator:
  1. **Clear Documentation:** It provides complete informal documentation with precise mathematical expressions and constraints.
  2. **Formal Specification:** It uses a formal language (WhyML) to create a mathematically sound, provable, and executable specification that eliminates ambiguity.
  3. **Reference Implementation:** It provides a simple, traceable C implementation generated from the formal spec, which can be used as a "golden reference" or oracle for testing.

#### Challenge 2: Lack of Traceability and Hidden Behaviors

* **The Problem:** A certification process demands clear traceability from requirements through design to implementation. Standard ONNX features like dynamic shape inference (where tensor shapes are calculated automatically) and the use of default attribute values create implicit behaviors. These features hide complexity and make it difficult to trace exactly how the final model is structured, which complicates verification.
* **SONNX Solution:** SONNX enforces explicitness. It proposes removing features like shape inference and default values. This forces every aspect of the model's architecture and operator configuration to be clearly and explicitly stated in the MLMD file, ensuring what you see is exactly what you get and dramatically improving traceability.

#### Challenge 3: Verifying the Final Implementation

* **The Problem:** The certification process ends with the integration and verification of the final avionics product. To do this effectively, the verification approach needs a perfect "oracle" to compare the final product against. A standard ONNX file, with its ambiguities, cannot reliably serve this purpose.
* **SONNX Solution:** By providing an executable formal specification and a reference C implementation, SONNX gives verification approach two powerful oracles. They can run the same test data through the final hardware/software and the SONNX reference implementations to prove that the behavior is identical, thereby satisfying a key certification objective.

These challenges mean that a standard ONNX file cannot serve as a reliable "single source of truth" for implementing a critical ML component. An engineer cannot guarantee that their implementation perfectly matches the designer's intent, which is a fundamental requirement for certification.

## Requirements for the SONNX Format

To overcome these challenges, the SONNX profile is being built to satisfy a stringent set of requirements derived from the needs of critical systems development.

### Guiding Principles as High-Level Requirements

The SONNX profile is guided by four main principles that act as top-level requirements:

1. **Clarity and Readability:** All specifications must be complete, unambiguous, and easily understandable.
2. **No Ambiguity:** The semantics of operators and the computation graph must be formally and precisely defined.
3. **Determinism:** The model's behavior and its demands on hardware resources (e.g., memory) must be predictable.
4. **Compatibility:** The profile must remain compatible with the broader ONNX standard, acting as a stricter subset rather than a completely new format.

### Detailed Technical Requirements

These principles translate into specific technical requirements for the format's components:

#### For Operators:

Each operator within the SONNX subset must be defined by three components:

* **Unambiguous Informal Documentation:** A requirement for a complete description with mathematical expressions, illustrations, and explicit statements of all constraints and restrictions (e.g., tensor shapes, attribute relationships).
* **Executable Formal Specification:** A requirement to provide a mathematically sound specification in a formal language (WhyML) that is both provable and executable, allowing it to serve as a **verification oracle** against which an implementation can be tested.
* **Traceable Reference Implementation:** A requirement for a simple C code implementation generated directly from the formal specification to serve as an additional, easily understood reference for verification.

#### For the Graph Model:

* **Mandatory Static Typing:** To eliminate the ambiguity of shape inference, all tensor shapes must be explicitly and statically defined throughout the graph.
* **No Implicit Values:** The use of default attribute values for operators is prohibited; all attributes must be explicitly stated in the model file.
* **Deterministic Semantics:** The execution semantics of the graph must be fully defined, removing any reliance on runtime-specific interpretations.

#### For Numerical Accuracy:

* The SONNX profile recognizes the difficulty of pre-specifying numerical accuracy for complex ML models. Therefore, the requirement is not for a fixed precision, but for the profile to provide a defined methodology and associated tools for estimating numerical errors, allowing developers to analyze and justify the accuracy of their specific implementation.

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 