Request to form Working Groups under ONNX

Probabilistic Models and ONNX

Proposal for ONNX steering Committee Review 12/03/25

What is Probabilistic Programming

Probabilistic Programming provides a unified framework for specifying statistical models and performing Bayesian inference automatically.

Key Concepts:

- Probabilistic Programming allows rigorous modelling of **noise, uncertainty, causal structure** and **latent variables**
- Models are specified using **random variables** with explicit probability distributions.
- The system automatically computes **log-likelihoods**, **posterior distributions**, and **uncertainty estimates**.
- Inference algorithms (HMC, NUTS, SMC, VI, Laplace, INLA) find distributions over parameters, not just point estimates.
- Probabilistic programming languages (PPLs) act as **compilers for Bayesian inference**, separating *model specification* from *inference execution*.

Why would this be beneficial in ONNX?

Extends ONNX beyond deterministic ML

- Adds native support for uncertainty-aware AI models and statistical reasoning.
- Complements GenAI and deterministic neural networks with **probabilistic inference**.

Unifies fragmented ecosystems

- Today: every PPL uses its own math kernels, RNG, distribution library, and inference implementation.
- With ONNX: one standardized IR for probabilistic computation across frameworks and devices.

Enables consistent, portable inference

- ONNX Runtime can execute Bayesian models across CPUs, GPUs, cloud, edge, and accelerators.
- Removes the need for framework-specific runtime engines.

Encourages hardware vendor adoption

- RNG, special functions, and inference kernels become standard targets.
- Vendors can optimize a stable, shared operator set instead of bespoke PPL internals.

Builds foundations for next-generation AI systems

- Combines symbolic/statistical modeling with modern ML.
- Supports model compression, distillation, and hybrid workflows.

What problem does this solve within this space

1. No standard IR for probabilistic models

Today each PPL (Stan, PyMC, Pyro, TFP, NumPyro, Turing.jl) emits its own internal representation for sampling and inference.
→ Hard to interoperate, optimize, or share models.

2. Inconsistent RNG semantics across frameworks

Each ecosystem has different:

- RNG behavior
- Seed handling
- Threading behavior
  → Breaks reproducibility across hardware and platforms.

3. Missing special functions and bijectors

These must be re-implemented per-framework.
→ Leads to numerical mismatch, drift, and portability challenges.

4. No shared accelerator story

PPLs rely on:

- Python
- Custom C++ kernels
- JAX-only acceleration
  → No unified path to devices, embedded systems, or vendor optimization.

5. No portable inference

Inference engines (HMC, NUTS, SMC, INLA, Pathfinder, Laplace) are tied to specific frameworks.
→ ONNX can become the **universal runtime** for Bayesian inference.

How this aligns with the direction of ONNX

1. Extends ONNX without breaking its design principles

- Operators remain **pure, functional, composable**.
- RNG is stateless and splittable → compatible with ONNX's execution graph model.
- Probabilistic ops fit naturally alongside NN, transformer, and GenAI workloads.

2. Complements existing ONNX Runtime strengths

- ORT already optimized for:
  - GPU/accelerator hardware
  - Parallelism
  - Subgraph execution
- These are the exact needs of MCMC, SMC, and Laplace/Pathfinder inference.

3. Mirrors ONNX GenAI extension strategy

- Similar to the GenAI opset effort:
  - Introduces well-scoped new operator domains
  - Minimizes changes to the ONNX core
  - Provides a roadmap for ecosystem adoption

4. Creates a cohesive mathematical foundation

- Special functions, distributions, and bijectors become shared assets.
- Hardware vendors gain a consistent target for acceleration.

5. Supports ONNX 2.0 Vision

- Probabilistic modeling becomes a **pillar** alongside deterministic ML and GenAI.
- Enables "hybrid" models combining neural networks + Bayesian inference pipelines.

What frameworks are we looking to support

We intend to support **major probabilistic programming frameworks**, starting with the most widely adopted:

First set of exporters:

- **Stan**
- **PyMC**

Next set of exporters to be considered once first set are completed:

- **NumPyro**
- **Pyro**
- **TensorFlow Probability**
- **JAX-native probabilistic models**
- **BayesFlow**

Future Framework Alignment

- **R-INLA**
- **Turing.jl**
  - Julia's primary probabilistic programming system
  - Alignment planned with Bijectors.jl and Distributions.jl
  - Enabled once ONNX bijector + distribution catalogs are complete

Goal:
Create a unified ONNX-backed probability and inference layer that all frameworks can target.

ONNX-Bayes Working Groups

Purpose:
Define how ONNX infrastructure, operators, and runtime constructs can support **probabilistic graphical modeling, Bayesian inference**, and **probabilistic programming languages (PPLs)**.

Scope:

- Introduce a **standardized probabilistic operator domain** (`ai.onnx.prob*`)
- Ensure **cross-framework compatibility** (Stan / PyMC / Pyro / NumPyro / TFP / JAX / BayesFlow)
- Provide **device-portable RNG semantics**, distributions, bijectors, and inference operators
- Reduce reliance on framework-specific probabilistic libraries or contrib ops
- Enable backend vendors to support probabilistic models **without reinventing RNG or special-function kernels**
- Position probabilistic inference as a first-class ONNX workload, complementary to GenAI, ML and Deep Learning

Requester : Brian Parbhu, Adam Pocock, Andreas Fehlner

Probabilistic Modeling & Inference WGs

Objective:
Define and standardize missing constructs required for probabilistic modeling and Bayesian inference inside ONNX.

WG Responsibilities:

- Define **new probabilistic operator domains** (`ai.onnx.prob`, `.special`, `.bijectors`, `.infer`)
- Collaborate with the Operators, Runtime, and Converter WGs
- Maintain a **canonical set of distributions, bijectors, and special functions**
- Ensure **RNG standardization** across runtimes using a splittable, stateless model
- Publish **reference implementations** for ONNX (CPU/CUDA)
- Provide guidance for exporter authors (Stan, PyMC and future ones i.e. Pyro, NumPyro, TFP, JAX, BayesFlow, R-INLA and Turing.jl)

Requester : Brian Parbhu, Adam Pocock, Andreas Fehlner

ONNX-Bayes Working Group 1

Probabilistic Operators & Functions

- Standardize:
- Some operators will need to be primitives and others will be functions
    - RNG: `SplitPRNG`, updated Random ops
    - Distributions (Normal → Dirichlet; Mixtures; HMMs)
    - Bijectors (constrained parameter transforms, flow layers)
    - Special functions (LogGamma, Digamma, Bessel, incomplete gamma, etc.)
- Publish decomposable **FunctionProto** reference graphs
- Ensure converter + backend alignment
- Reference CPU/CUDA kernels for ORT

Requester : Brian Parbhu , Adam Pocock, Andreas Fehlner

ONNX-Bayes Working Group 2 (Inference and Pipelines)

Standardize inference operators:

- **Laplace**, **Pathfinder**, **INLA**, **HMC**, **NUTS**, **Gibbs**, **SMC**

Define end-to-end probabilistic pipelines: sampling, diagnostics, SBC

Vendor integration via execution-provider interfaces

Explore **accelerator-friendly inference** (warp-aware NUTS, batched MCMC, vectorized SMC)

Requester : Brian Parbhu, Adam Pocock, Andreas Fehlner

ONNX-Bayes Working Group 3 (Exporters & Framework Alignment)

Standardize conversion patterns for:

- **Stan**, **PyMC**, **Pyro**, **NumPyro**, **TFP**, **JAX**, **BayesFlow**
- **Future: R-INLA and Turing.jl** (R and Julia ecosystem alignment)

Provide scripts + methods to upgrade existing PPL workflows to ONNX

Maintain cross-framework shape, mask, and plate semantics

Technical Deliverables (Operators, Semantics, Runtime)

Operator-Level Deliverables

- RNG: Seedable, reproducible, stateless, splittable PRNG semantics
- Probabilistic Domain: `LogProb`, `Observe`, `Factor`, `Random*`
- Distribution & Bijector Catalogs
- Special Function Domain: Gamma family, Bessel family, Erfc/ErfInv, etc.

Inference Deliverables

- Approximate inference: Laplace, Pathfinder, INLA
- MCMC: HMC, NUTS, Slice, Gibbs
- SMC: tempered, variational-guided, and resampling layers
- Diagnostics: ESS, R-hat, divergences, SBC

Runtime Deliverables

- ONNX Runtime extension library (`onnxruntime-prob-extensions`)
- GPU-accelerated special functions + inference kernels
- Subgraph APIs for log-joint invocation
- Reproducible multi-thread and multi-device execution

Requester : Brian Parbhu , Adam Pocock, Andreas Fehlner

Next Steps & WGs Formation Plan

Next Steps

- Present full proposal to ONNX Steering Committee
- Approve formation of **Probabilistic Modeling & Inference WGs**
- Charter Working Groups (Operators, Inference Pipelines, Exporters)
- Establish dedicated ONNX GitHub repos for specs + reference code
- Begin bi-weekly working meetings
- Coordinate with Runtime, Operators, Model Zoo, Converters WGs

Initial Deliverables After Approval

- RNG specification + base probabilistic ops
- Special functions + first distributions
- First set of exporters: Stan, PyMC ,
- Future exporter MVP considerations i.e. pyro, numpyro, Tensorflow Probability, jax-based frameworks
- Draft opset for `ai.onnx.prob*` domains

Longer-Term Targets

- Full probabilistic opset v1
- Complete exporter suite
- GPU-optimized NUTS / SMC
- **Turing.jl → ONNX interoperability**
- **R-INLA → ONNX interoperability**
- Integration with broader ONNX 2.0 roadmap