BEAST 3: What's New for Package Developers

BEAST Core Developer Team

2026

Outline

The Big Picture

Three pillars of change in BEAST 3:

1. **Maven + JPMS** — modern build system and module isolation
2. **Strongly-typed spec system** — compile-time checked inputs and domain types
3. **New package distribution** — Maven Central JARs, module-based service discovery

Goal

Catch more errors at compile time, simplify dependency management, and make packages first-class citizens in the Java module system.

Ant → Maven

BEAST 2

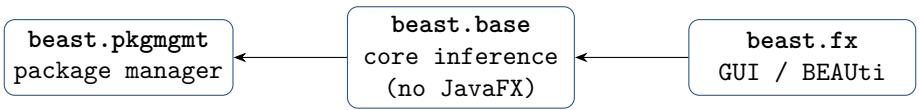
- Ant `build.xml`
- Manual JAR management
- One monolithic project
- Classpath-based

BEAST 3

- Maven `pom.xml`
- Automatic dependency resolution
- Multi-module project
- Module-path-based (JPMS)

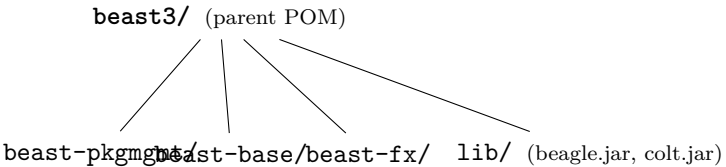
Java 25 + JPMS Modules

- Requires **JDK 25** (was Java 8/11)
- JavaFX is a **Maven dependency** — no bundled JDK needed
- Three JPMS modules, each with `module-info.java`:



All declared as `open module` (reflection OK for XML unmarshalling).

Project Structure



- `beast-base` has **no JavaFX dependency** — can be used headlessly
- `beagle` and `colt` ship as modular JARs in `lib/`

What This Means for Your Package

1. **Recommended:** create a Maven `pom.xml` for your package
   - See `beast-package-skeleton` on GitHub
2. **Alternative:** keep Ant, but update classpath JARs
3. Add a `module-info.java` with `provides` declarations
4. Target **Java 25** (use `-release 25`)

Tip

The `beast-package-skeleton` repo is a ready-to-copy Maven project with all of this pre-configured.

Why Types?

Problem: In BEAST 2, almost everything is `Input<RealParameter>` or `Input<Function>`

- Type errors discovered at **runtime** (wrong parameter passed to wrong input)
- Bounds (`lower="0"`) are stringly-typed XML attributes
- No way to express "this must be a positive real" in the Java API

Solution: New `beast.base.spec` hierarchy

- Compile-time checked typed inputs
- Domain types encode valid ranges
- Self-documenting model structure

Parameter Types

`RealParameter` is replaced by specific types:

BEAST 2	**BEAST 3**	**Use for**
`RealParameter` (dim=1)	`RealScalarParam`	single real value
`RealParameter` (dim>1)	`RealVectorParam`	vector of reals
`IntegerParameter`	`IntScalarParam` / `IntVectorParam`	integer values
`BooleanParameter`	`BoolScalarParam` / `BoolVectorParam`	boolean values
`RealParameter` (simplex)	`SimplexParam`	probability simplex

Domain Types

Domains replace explicit `lower`/`upper` bounds:

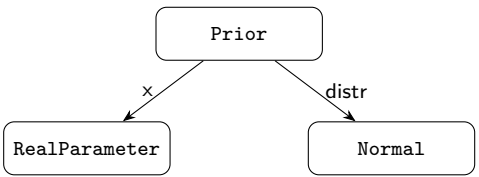
BEAST 2 XML pattern	BEAST 3 domain
`RealParameter lower="0"`	`PositiveReal.INSTANCE`
`RealParameter lower="0" upper="1"`	`UnitInterval.INSTANCE`
`RealParameter` (unbounded)	`Real.INSTANCE`
`IntegerParameter lower="0"`	`NonNegativeInt.INSTANCE`
`IntegerParameter lower="1"`	`PositiveInt.INSTANCE`

Domains propagate through the model graph at initialization time.

"Distribution IS the Prior"

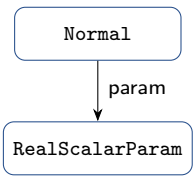
BEAST 2

`Prior` wraps distribution + parameter:



BEAST 3

Distribution **is** the prior:



No separate `Prior` wrapper needed — each distribution has a `param` input directly (from `TensorDistribution`).

Spec Classes Coexist with Legacy

- Legacy classes are **deprecated**, not removed
- Each deprecated class links to its spec replacement:

```
/**
 * @deprecated use {@link RealScalarParam} or {@link RealVectorParam}
 */
public class RealParameter extends ...
```

- Your IDE shows deprecated usage (strikethrough)
- **Migrate incrementally** — old and new classes can coexist
- Consider a `yourpackage.spec` sub-package for backward compatibility

Before / After: Input Declaration

BEAST 2

```
public Input<RealParameter> kappaInput =
    new Input<>("kappa",
    "ts/tv ratio",
    Validate.REQUIRED);
```

BEAST 3

```
public Input<RealScalar<PositiveReal>>
    kappaInput = new Input<>("kappa",
    "ts/tv ratio",
    Validate.REQUIRED);
```

- Domain (`PositiveReal`) is part of the type — no runtime `lower="0"` needed
- Compiler catches if someone passes a `UnitInterval` parameter where `PositiveReal` is expected

When NOT to Use Spec Types

Spec parameters are for MCMC-estimated quantities only.

Keep plain `Input<Integer>`, `Input<Double>`, etc. for:

- Dimension counts
- Category numbers
- Boolean flags
- Configuration / initializer values

Rule of thumb

If it appears in the MCMC state, use a spec parameter type.
If it's fixed configuration, use a plain `Input<>`.

New Distributions

Genuinely new — not present in BEAST 2:

Continuous

- `Cauchy` — location / scale
- `Laplace` — μ / scale
- `InverseGamma` — α / β
- `ChiSquare` — degrees of freedom
- `GammaMean` — mean parameterization

Discrete & Wrappers

- `Bernoulli` — boolean with prob p
- `IntUniform` — discrete uniform
- `IID` — scalar dist $\rightarrow$ vector
- `OffsetReal` — additive offset
- `TruncatedReal` — truncation bounds

New / Refactored Operators

- **`IntervalScaleOperator`** (new) — scales intervals between consecutive node heights; preserves topology, changes relative branch lengths
- **`ScaleTreeOperator`** (split) — dedicated tree-scaling operator, separated from the parameter `ScaleOperator`
- **Spec `UpDownOperator`** — `up`/`down` take `List<Scalable>`; uses Bactrian kernel by default

Simplified Default Tree Operator Set

BEAST 2 (BICEPS)

- `EpochFlexOperator` (top)
- `EpochFlexOperator` (all)
- `TreeStretchOperator`
- `BactrianScaleOperator`
- `BactrianNodeOperator`
- `BactrianSubtreeSlide`
- `Exchange` (narrow/wide)
- `WilsonBalding`

BEAST 3

- **`UpDownOperator`** (replaces 3 BICEPS ops)
- `BactrianScaleOperator`
- `BactrianNodeOperator`
- `BactrianSubtreeSlide`
- `Exchange` (narrow/wide)
- `WilsonBalding`

StateNode Changes

Breaking

`StateNode` is **no longer a** `Function`.

- If your code relies on `StateNode` implementing `Function`, explicitly implement `Function` yourself (intermediate step) or migrate to typed spec parameters

Breaking

`StateNode.scale()` has been **removed**.

- Implement the `Scalable` interface if your `StateNode` needs scaling support

Removed Classes

Removed	Replacement
`beast.base.inference.Evaluator`	(removed — impacted MCMC subclasses)
`AscertainedAlignment`	Use standard `Alignment`

- Check your code for imports of these classes
- `Evaluator` mainly affected custom MCMC acceptance logic

Narrowed Operator Input Types

Spec operators accept **typed** parameters:

- `ScaleOperator.parameter` → `Scalable` interface
- `RealRandomWalkOperator` → `RealVectorParam` xor `RealScalarParam`
- `BitFlipOperator` → `BoolVectorParam`
- `DeltaExchangeOperator` — adds typed inputs (`rvparameter`, `ivparameter`, etc.)

See the migration guide for the complete operator mapping table.

Service Discovery: `module-info.java`

Primary mechanism — BEAST scans module descriptors at startup:

```
open module my.beast.package {
    requires beast.pkgmgmt;
    requires beast.base;

    provides beast.base.core.BEASTInterface with
        my.beast.package.MyModel,
        my.beast.package.MyOperator;
}
```

- Works in both the boot layer (IDE development) and plugin layers (deployed packages)
- IntelliJ/Eclipse discover services automatically when both projects are open

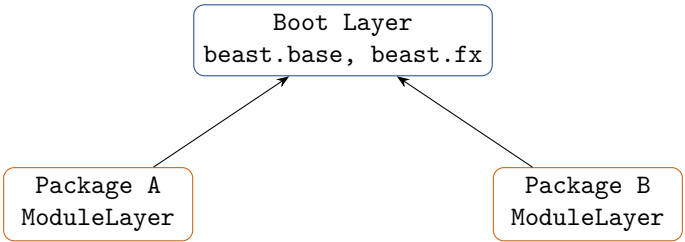
Service Discovery: `version.xml` Fallback

- **Still works** for non-modular JARs
- JARs without `module-info` are treated as automatic modules
- Services registered from `version.xml` as before
- Useful for gradual migration

Recommendation

Add `module-info.java` when you can — it's the primary mechanism and gives you compile-time module dependency checking.

Plugin ModuleLayer Isolation



- Each deployed package gets its **own ModuleLayer**
- No classpath conflicts between packages
- Packages can have different versions of shared dependencies

Maven Central Distribution

New: packages can be published as plain Maven Central JARs.

- Users install via BEAUti: **Install from Maven** button
- Or CLI: `packagemanager -maven groupId:artifactId:version`
- Resolution via Apache Maven Resolver
- JARs cached in `~/.beast/2.8/maven-repo/`
- Tracked in `maven-packages.xml`

- Custom repos: `packagemanager -addMavenRepository <url>`
- ZIP packages still work alongside Maven packages

beast-package-skeleton

Ready-to-copy template project on GitHub:

- Maven `pom.xml` with BEAST 3 dependencies
- `module-info.java` with `provides` declarations
- Example `ScalarDistribution`, operator, and tests
- BEAST XML using the strongly-typed spec API
- Maven Central release profile (source, javadoc, GPG, publishing plugin)
- `version.xml` JAR embedding

Start here: `https://github.com/CompEvol/beast-package-skeleton`

Testing Upgrades

JUnit 5

- Primary framework (Jupiter 5.8.2)
- JUnit 4 still compiles via vintage engine
- `@Tag("slow")` for long MCMC tests
- `mvn test` — fast tests only
- `mvn test -Pslow-tests` — all

TestFX (GUI testing)

- BEAUti tests run **headlessly**
- Monocle Glass platform
- No display server needed
- TestFX 4.0.18 + JUnit 5 integration

Dependency Updates

Dependency	BEAST 2	BEAST 3
Java	8 / 11	25
Build	Ant	Maven 3.9+
Math	commons-math3	commons-math4-legacy 4.0 + commons-numbers / rng / statistics
ANTLR	4.x (bundled)	4.13.2 (Maven)
JavaFX	bundled with JDK	25.0.2 (Maven)
JUnit	4	5 (Jupiter 5.8.2)
GUI tests	—	TestFX 4.0.18

BEAUti & Templates

- **fxtemplates namespacing** — templates moved to module-unique paths (`beast.fx/fxtemplates/`) to avoid JPMS split-package conflicts
- **Module resource scanning** — `BeautiDoc` scans JPMS module resources, so templates are discovered from JARs on the module path
- External packages use their own namespace:
  e.g. `beast.morph.models.fx/fxtemplates/`
- Legacy path (`fxtemplates/*.xml`) still supported as fallback

Migration Checklist

1. **Update build** — create `pom.xml` (or update Ant classpath JARs), target Java 25
2. **Fix compile errors** — `StateNode` not `Function`, `scale()` removed, `Evaluator` gone
3. **Add `module-info.java`** — declare `provides` for your `BEASTInterface` implementations
4. **Migrate to spec types** — replace `RealParameter` inputs with `RealScalarParam` / `RealVectorParam` and domain types
5. **Update XML & templates** — use spec class names, new operator set, namespaced fxtemplates

Resources

Migration guide `scripts/migration-guide.md` — class mapping tables, step-by-step instructions

CHANGELOG `CHANGELOG.md` — full list of what changed and why

Package skeleton `https://github.com/CompEvol/beast-package-skeleton`

BEAST 3 repo `https://github.com/CompEvol/beast3`