

TracNav

- [JPFWiki](#) - Welcome Page

[Introduction...](#)

[Installing JPF...](#)

[User Guide...](#)

[Developer Guide...](#)

[Projects](#)

- [jpf-core](#)
 - [jpf-actor](#)
 - [jpf.awt](#)
 - [jpf.awt-shell](#)
 - [jpf-concurrent](#)
 - [jpf-cv](#)
 - [jpf-delayed](#)
 - [jpf-guided-test](#)
 - [jpf-mango](#)
 - [jpf-racefinder](#)
 - [jpf-rtembed](#)
 - [jpf-statechart](#)
 - [net-iocache](#)
 - [jpf-aprop](#)
 - [jpf-numeric](#)
 - [jpf-symbc](#)
 - [jpf-concolic](#)
 - [jpf-symbc-load?](#)
 - [jpf-extended-test-gen](#)
 - [jpf-parallel-spf?](#)
 - [eclipse-jpf](#)
 - [netbeans-jpf](#)
 - [jpf-inspector](#)
 - [jpf-shell](#)
 - [jpf-template](#)
 - [jpf-trace-server](#)
 - [standard NB example](#)
- [Summer Projects](#)
 - [External Projects](#)
 - [Change\(B\)log](#)

[About...](#)

- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

jpf-core

This is the basis for all JPF projects, i.e. you always need to install it. jpf-core contains the basic VM and model checking infrastructure, and can be used to check for concurrency defects like deadlocks, and unhandled exceptions like NullPointerExceptions and AssertionErrors.

Repository

The Mercurial repository is on <http://babelfish.arc.nasa.gov/hg/jpf/jpf-core>

Properties

jpf-core supports two rather generic properties, which are configured by default:

- gov.nasa.jpf.jvm.NoUncaughtExceptionsProperty
- gov.nasa.jpf.jvm.NotDeadlockedProperty

There is no need to parameterise any of them. NoUncaughtExceptionsProperty covers all Throwables that are not handled within the application, i.e. would terminate the process.

A number of the listeners (like PreciseRaceDetector) are ListenerAdapter instances, i.e. work as more specific Property implementations.

Listeners

jpf-core includes a variety of [listeners](#) that fall into three major categories:

- program properties
- execution monitoring
- execution control

The main listeners are

- [AssertionProperty](#)
- [BudgetChecker?](#)
- [CGMonitor?](#)
- [TraceStorer?](#)
- [ChoiceSelector?](#)
- [CoverageAnalyzer?](#)
- [DeadlockAnalyzer?](#)
- [ExecTracker?](#)
- [IdleFilter](#)
- [PathOutputMonitor?](#)
- [PreciseRaceDetector?](#)
- [ExceptionInjector](#)
- [SearchMonitor?](#)
- [LogConsole?](#)

A complete list of listeners available can be found [here](#).

Properties

jpf-core uses many JPF properties, you can find most of them in the defaults.properties file. The following ones are of interest for users

- **listener** - a comma separated list of class names with listeners that should be automatically instantiated and registered during JPF startup
- **listener.autoload** - a comma separated list of annotation types. If JPF encounters such an annotation in one of the analyzed classes at runtime, it automatically loads and registered the associated listener
- **listener.<annotation-type>** - class name of the listener associated with <annotation-type>
- **vmInsn_factory.class** - class name of a [BytecodeInstructionFactory](#), e.g. to switch to symbolic execution mode or to use specific bytecode implementations for checking numeric properties

- **vm.halt_on_throw** (true|false) - tells JPF if it should try to find a handler if it encounters an exception in the analyzed program (useful to avoid masking exceptions within handlers)
 - **cg.randomize_choices** (random|path|def) - tells JPF if it should randomize the order of choices for each [ChoiceGenerator](#), to avoid degenerated searches (e.g. always starting with the main thread in scheduling choices).
- report.console.propertyViolation** - comma separated list of topics that should be printed by JPF if it detects an error. Possible values include
- `error` error description
 - `snapshot` thread stacks
 - `trace` instruction/statement trace (can be long and memory-expensive)