Minimal Value Types Command-line Options

This page describes the command line for the experimental java compiler, the java launcher and HotSpot for the support of Minimal Value Types in the valhalla project.

Minimal Value Types flags:

	Flag	Description
hotspot		
REQUIRED	**-XX:EnableMVT**	Boolean value, default value is `false` If true enable support for minimal value types with a value capable class and an associated derived value type.
REQUIRED	**-XVerify:none**	Disable verification. Verification is not yet supported.
	-XX:ValueArrayFlatten	Boolean value, default value is `true` Flatten arrays of values, if possible
	-XX:ValueArrayElemMaxFlatSize	Integer value, default is -1 Max size for flattening an array of values, < 0 means no limit
	-XX:ValueArrayElemMaxFlatOops	Integer value, default is 4 Max number of embedded object references in value type to flatten in an array, < 0 means no limit
	-XX:BigValueTypeThreshold	size_t value in bytes, default is 4 * BytesPerLong Max value type size for interpreter buffering of local variable table entries
	-XX:ValueTypesBufferMaxMemory	Integer value, default 128 (pages) Max memory used for value type buffers
	-XX:ValueTypesThreadLocalRecycling	Boolean value, default value is 'true' Enable thread local recycling of buffered values in the interpreter
	-XX:MinimumVTBufferChunkPerFrame	Integer value, default 2 Minimum number of VT buffer chunks allowed per frame
	-XX:ReportVTBufferRecyclingTimes	Boolean value, default false Print duration of each VTBuffer recycling for the interpreter
	-XX:ValueTypePassFieldsAsArgs	Non-product: Boolean value, default value is `true` Pass each value type field as an argument to a method call instead of a value type reference
	-XX:ValueTypeReturnedAsFields	Non-product: Boolean value, default value is `true` Return value type fields instead of a value type reference
	-XX:ValueArrayAtomicAccess	Boolean value, default value is `false` Enable atomic access to values in an array, by treating value types as references.
	-XX:-UseTLAB	To see heap allocations which do not use optimized Thread Local Allocation Buffers
	-XX:+PrintEliminateAllocations	Non-product builds: Print out when allocations are eliminated
	-XX:+PrintEscapeAnalysis	Non-product builds: Print results of escape analysis (e.g. if you believe boxing was not eliminated)
java		
	-Djava.lang.invoke.MethodHandle.DUMP_CLASS_FILES	Boolean value, default value is `false` If true dump class files generated for lambda forms and derived value types into a directory named DUMP_CLASS_FILES under the current directory.
	-Dvalhalla.enableValueLambdaForms	Boolean value, default value is `true` If true then lambda forms with value types in their signature will use the Q type __Value and value-type specific byte code will be generated.

	-Dvalhalla.dumpProxyClasses	Boolean value, default value is `false` If true, dumps MVT lambda forms.
	-Dvalhalla.enablePoolPatches	Boolean value, default value is `false` If true and if "valhalla.enableValueLambdaForms" is true then lambda forms with value types in their signature will generate byte code with constant pool patching where appropriate.
javac		
	-Xlint:values	Lint category for warnings associated to bad usage of value capable classes - examples: • VCC not final • VCC cannot 'extend' another class • overrides of 'bad' Object methods (such as wait/notify) • illegal modifiers on VCC such as 'synchronized' • non-final instance fields in VCC • cannot assign 'null' to variable of VCC type