//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#ifdef PHASE

PHASE(All)

PHASE(BGJit)

PHASE(LibInit)

PHASE(JsLibInit)

PHASE(Parse)

PHASE(RegexCompile)

PHASE(DeferParse)

PHASE(ParserBind)

PHASE(FunctionSourceInfoParse)

PHASE(StringTemplateParse)

PHASE(SkipNestedDeferred)

PHASE(CacheScopeInfoNames)

PHASE(ScanAhead)

PHASE(ParallelParse)

PHASE(EarlyReferenceErrors)

PHASE(ByteCode)

PHASE(CachedScope)

PHASE(StackFunc)

PHASE(StackClosure)

PHASE(DelayCapture)

PHASE(DebuggerScope)

PHASE(ByteCodeSerialization)

PHASE(VariableIntEncoding)

PHASE(NativeCodeSerialization)

PHASE(Delay)

PHASE(Speculation)

PHASE(GatherCodeGenData)

PHASE(Asmjs)

PHASE(AsmjsTmpRegisterAllocation)

PHASE(AsmjsEncoder)

PHASE(AsmjsInterpreter)

PHASE(AsmJsJITTemplate)

PHASE(AsmjsFunctionEntry)

PHASE(AsmjsInterpreterStack)

PHASE(AsmjsEntryPointInfo)

PHASE(BackEnd)

PHASE(IRBuilder)

PHASE(SwitchOpt)

PHASE(BailOnNoProfile)

PHASE(BackendConcatExprOpt)

PHASE(ClosureRangeCheck)

PHASE(ClosureRegCheck)

PHASE(Inline)

PHASE(InlineRecursive)

PHASE(InlineAtEveryCaller) //Inlines a function, say, foo at every caller of foo. Doesn't guarantee all the calls within foo are inlined too.

PHASE(InlineTree) //Inlines every function within a top function, say, foo (which needs to be top function) Note: -force:inline achieves the effect of both -force:InlineTree & -force:InlineAtEveryCaller

PHASE(TryAggressiveInlining)

PHASE(InlineConstructors)

PHASE(InlineBuiltIn)

PHASE(InlineInJitLoopBody)

PHASE(InlineAccessors)

PHASE(InlineGetters)

PHASE(InlineSetters)

PHASE(InlineApply)

PHASE(InlineApplyTarget)

PHASE(BailOutOnNotStackArgs)

PHASE(InlineCall)

PHASE(InlineCallTarget)

PHASE(PartialPolymorphicInline)

PHASE(PolymorphicInline)

PHASE(PolymorphicInlineFixedMethods)

PHASE(InlineOutsideLoops)

PHASE(InlineFunctionsWithLoops)

PHASE(EliminateArgoutForInlinee)

PHASE(InlineBuiltInCaller)

PHASE(InlineArgsOpt)

PHASE(RemoveInlineFrame)

PHASE(InlinerConstFold)

PHASE(ExecBOIFastPath)

PHASE(FGBuild)

PHASE(RemoveBreakBlock)

PHASE(TailDup)

PHASE(FGPeeps)

PHASE(GlobOpt)

PHASE(OptimizeTryCatch)

PHASE(Backward)

PHASE(TrackIntUsage)

PHASE(TrackNegativeZero)

PHASE(TypedArrayVirtual)

PHASE(TrackIntOverflow)

PHASE(TrackCompoundedIntOverflow)

PHASE(Forward)

PHASE(ValueTable)

PHASE(PathDependentValues)

PHASE(TrackRelativeIntBounds)

PHASE(BoundCheckElimination)

PHASE(BoundCheckHoist)

PHASE(LoopCountBasedBoundCheckHoist)

PHASE(CopyProp)

PHASE(ObjPtrCopyProp)

PHASE(ConstProp)

PHASE(ConstFold)

PHASE(CSE)

PHASE(HoistConstInt)

PHASE(TypeSpec)

PHASE(AggressiveIntTypeSpec)

PHASE(AggressiveMulIntTypeSpec)

PHASE(LossyIntTypeSpec)

PHASE(FloatTypeSpec)

PHASE(StringTypeSpec)

PHASE(InductionVars)

PHASE(Invariants)

PHASE(FieldCopyProp)

PHASE(FieldHoist)

PHASE(FieldPRE)

PHASE(HostOpt)

PHASE(ObjTypeSpec)

PHASE(ObjTypeSpecNewObj)

PHASE(ObjTypeSpecIsolatedFldOps)

PHASE(ObjTypeSpecIsolatedFldOpsWithBailOut)

PHASE(ObjTypeSpecStore)

PHASE(EquivObjTypeSpec)

PHASE(EquivObjTypeSpecByDefault)

PHASE(TraceObjTypeSpecTypeGuards)

PHASE(TraceObjTypeSpecWriteGuards)

PHASE(LiveOutFields)

PHASE(DisabledObjTypeSpec)

#if DBG

PHASE(SimulatePolyCacheWithOneTypeForFunction)

#endif

PHASE(CheckThis)

PHASE(StackArgOpt)

PHASE(IndirCopyProp)

PHASE(ArrayCheckHoist)

PHASE(ArrayMissingValueCheckHoist)

PHASE(ArraySegmentHoist)

PHASE(JsArraySegmentHoist)

PHASE(ArrayLengthHoist)

PHASE(EliminateArrayAccessHelperCall)

PHASE(NativeArray)

PHASE(NativeNewScArray)

PHASE(NativeArrayConversion)

PHASE(CopyOnAccessArray)

PHASE(NativeArrayLeafSegment)

PHASE(TypedArrayTypeSpec)

PHASE(LdLenIntSpec)

PHASE(FixDataProps)

PHASE(FixMethodProps)

PHASE(FixAccessorProps)

PHASE(FixDataVarProps)

PHASE(UseFixedDataProps)

PHASE(UseFixedDataPropsInInliner)

PHASE(LazyBailout)

PHASE(LazyFixedDataBailout)

PHASE(LazyFixedTypeBailout)

PHASE(FixedMethods)

PHASE(FEFixedMethods)

PHASE(FixedFieldGuardCheck)

PHASE(FixedNewObj)

PHASE(JitAllocNewObj)

PHASE(FixedCtorInlining)

PHASE(FixedCtorCalls)

PHASE(FixedScriptMethodInlining)

PHASE(FixedScriptMethodCalls)

PHASE(FixedBuiltInMethodInlining)

PHASE(FixedBuiltInMethodCalls)

PHASE(SplitNewScObject)

PHASE(OptTagChecks)

PHASE(MemOp)

PHASE(MemSet)

PHASE(MemCopy)

PHASE(DeadStore)

PHASE(ReverseCopyProp)

PHASE(MarkTemp)

PHASE(MarkTempNumber)

PHASE(MarkTempObject)

PHASE(MarkTempNumberOnTempObject)

PHASE(Lowerer)

PHASE(FastPath)

PHASE(LoopFastPath)

PHASE(LeafFastPath)

PHASE(MathFastPath)

PHASE(Atom)

PHASE(MulStrengthReduction)

PHASE(AgenPeeps)

PHASE(BranchFastPath)

PHASE(CallFastPath)

PHASE(BitopsFastPath)

PHASE(OtherFastPath)

PHASE(ObjectFastPath)

PHASE(ProfileBasedFldFastPath)

PHASE(AddFldFastPath)

PHASE(RootObjectFldFastPath)

PHASE(ArrayLiteralFastPath)

PHASE(ArrayCtorFastPath)

PHASE(NewScopeSlotFastPath)

PHASE(FrameDisplayFastPath)

PHASE(HoistMarkTempInit)

PHASE(HoistConstAddr)

PHASE(PreLowererPeeps)

PHASE(CFGInJit)

PHASE(TypedArray)

PHASE(TracePinnedTypes)

PHASE(InterruptProbe)

PHASE(EncodeConstants)

PHASE(RegAlloc)

PHASE(Liveness)

PHASE(LinearScan)

PHASE(OpHelperRegOpt)

PHASE(StackPack)

PHASE(SecondChance)

PHASE(RegionUseCount)

PHASE(RegHoistLoads)

PHASE(ClearRegLoopExit)

PHASE(Peeps)

PHASE(Layout)

PHASE(EHBailoutPatchUp)

PHASE(FinalLower)

PHASE(PrologEpilog)

PHASE(InsertNOPs)

PHASE(Encoder)

PHASE(Emitter)

#if defined(\_M\_IX86) || defined(\_M\_X64)

PHASE(BrShorten)

PHASE(LoopAlign)

#endif

#ifdef RECYCLER\_WRITE\_BARRIER

#if DBG\_DUMP

PHASE(SWB)

#endif

#endif

PHASE(Run)

PHASE(Interpreter)

PHASE(EvalCompile)

PHASE(FastIndirectEval)

PHASE(IdleDecommit)

PHASE(IdleCollect)

PHASE(MemoryAllocation)

#ifdef RECYCLER\_PAGE\_HEAP

PHASE(PageHeap)

#endif

PHASE(LargeMemoryAllocation)

PHASE(PageAllocatorAlloc)

PHASE(Recycler)

PHASE(ThreadCollect)

PHASE(ExplicitFree)

PHASE(ExpirableCollect)

PHASE(GarbageCollect)

PHASE(ConcurrentCollect)

PHASE(BackgroundResetMarks)

PHASE(BackgroundFindRoots)

PHASE(BackgroundRescan)

PHASE(BackgroundRepeatMark)

PHASE(BackgroundFinishMark)

PHASE(ConcurrentPartialCollect)

PHASE(ParallelMark)

PHASE(PartialCollect)

PHASE(ResetMarks)

PHASE(ResetWriteWatch)

PHASE(FindRoot)

PHASE(FindRootArena)

PHASE(FindImplicitRoot)

PHASE(FindRootExt)

PHASE(ScanStack)

PHASE(ConcurrentMark)

PHASE(ConcurrentWait)

PHASE(Rescan)

PHASE(Mark)

PHASE(Sweep)

PHASE(SweepWeak)

PHASE(SweepSmall)

PHASE(SweepLarge)

PHASE(SweepPartialReuse)

PHASE(ConcurrentSweep)

PHASE(Finalize)

PHASE(Dispose)

PHASE(FinishPartial)

PHASE(Host)

PHASE(BailOut)

PHASE(RegexQc)

PHASE(InlineCache)

PHASE(PolymorphicInlineCache)

PHASE(MissingPropertyCache)

PHASE(CloneCacheInCollision)

PHASE(ConstructorCache)

PHASE(InlineCandidate)

PHASE(InlineHostCandidate)

PHASE(ScriptFunctionWithInlineCache)

PHASE(Arena)

PHASE(ApplyUsage)

PHASE(ObjectHeaderInlining)

PHASE(ObjectHeaderInliningForConstructors)

PHASE(ObjectHeaderInliningForObjectLiterals)

PHASE(ObjectHeaderInliningForEmptyObjects)

#if DBG\_DUMP

PHASE(TypePropertyCache)

PHASE(InlineSlots)

#endif

PHASE(DynamicProfile)

#ifdef DYNAMIC\_PROFILE\_STORAGE

PHASE(DynamicProfileStorage)

#endif

PHASE(JITLoopBody)

PHASE(JITLoopBodyInTryCatch)

PHASE(ReJIT)

PHASE(ExecutionMode)

PHASE(SimpleJitDynamicProfile)

PHASE(SimpleJit)

PHASE(FullJit)

PHASE(FailNativeCodeInstall)

PHASE(PixelArray)

PHASE(Etw)

PHASE(Profiler)

PHASE(CustomHeap)

PHASE(XDataAllocator)

PHASE(PageAllocator)

PHASE(StringConcat)

#if DBG\_DUMP

PHASE(PRNG)

#endif

PHASE(PreReservedHeapAlloc)

PHASE(CFG)

PHASE(ExceptionStackTrace)

PHASE(ExtendedExceptionInfoStackTrace)

PHASE(ProjectionMetadata)

PHASE(TypeHandlerTransition)

PHASE(Debugger)

PHASE(ENC)

PHASE(ConsoleScope)

PHASE(ScriptProfiler)

PHASE(JSON)

PHASE(RegexResultNotUsed)

PHASE(Error)

PHASE(PropertyRecord)

PHASE(TypePathDynamicSize)

PHASE(ConditionalCompilation)

PHASE(InterpreterProfile)

PHASE(InterpreterAutoProfile)

PHASE(ByteCodeConcatExprOpt)

PHASE(TraceInlineCacheInvalidation)

PHASE(TracePropertyGuards)

#ifdef ENABLE\_JS\_ETW

PHASE(StackFramesEvent)

#endif

PHASE(PerfHint)

PHASE(DeferSourceLoad)

PHASE(ObjectMutationBreakpoint)

#undef PHASE

#endif

#ifndef DEFAULT\_CONFIG\_BgJitDelay

#if \_M\_ARM

#define DEFAULT\_CONFIG\_BgJitDelay (70)

#else

#define DEFAULT\_CONFIG\_BgJitDelay (30)

#endif

#define DEFAULT\_CONFIG\_ASMJS (true)

#define DEFAULT\_CONFIG\_AsmJsEdge (false)

#define DEFAULT\_CONFIG\_AsmJsStopOnError (false)

#define DEFAULT\_CONFIG\_SIMDJS (false)

#define DEFAULT\_CONFIG\_BgJitDelayFgBuffer (0)

#define DEFAULT\_CONFIG\_BgJitPendingFuncCap (31)

#define DEFAULT\_CONFIG\_CurrentSourceInfo (true)

#define DEFAULT\_CONFIG\_CreateFunctionProxy (true)

#define DEFAULT\_CONFIG\_HybridFgJit (false)

#define DEFAULT\_CONFIG\_HybridFgJitBgQueueLengthThreshold (32)

#define DEFAULT\_CONFIG\_Prejit (false)

#define DEFAULT\_CONFIG\_DeferNested (true)

#define DEFAULT\_CONFIG\_DeferTopLevelTillFirstCall (true)

#define DEFAULT\_CONFIG\_DefineGetterSetter (true)

#define DEFAULT\_CONFIG\_DirectCallTelemetryStats (false)

#define DEFAULT\_CONFIG\_errorStackTrace (true)

#define DEFAULT\_CONFIG\_FastPathCap (-1) // By default, we do not have any fast path cap

#define DEFAULT\_CONFIG\_FastLineColumnCalculation (true)

#define DEFAULT\_CONFIG\_PrintLineColumnInfo (false)

#define DEFAULT\_CONFIG\_ForceDecommitOnCollect (false)

#define DEFAULT\_CONFIG\_ForceDeferParse (false)

#define DEFAULT\_CONFIG\_NoDeferParse (false)

#define DEFAULT\_CONFIG\_ForceDynamicProfile (false)

#define DEFAULT\_CONFIG\_ForceExpireOnNonCacheCollect (false)

#define DEFAULT\_CONFIG\_ForceFastPath (false)

#define DEFAULT\_CONFIG\_ForceJITLoopBody (false)

#define DEFAULT\_CONFIG\_ForceCleanPropertyOnCollect (false)

#define DEFAULT\_CONFIG\_ForceCleanCacheOnCollect (false)

#define DEFAULT\_CONFIG\_ForceGCAfterJSONParse (false)

#define DEFAULT\_CONFIG\_ForceSerialized (false)

#define DEFAULT\_CONFIG\_ForceES5Array (false)

#define DEFAULT\_CONFIG\_ForceAsmJsLinkFail (false)

#define DEFAULT\_CONFIG\_DumpCommentsFromReferencedFiles (false)

#define DEFAULT\_CONFIG\_ExtendedErrorStackForTestHost (false)

//Following determines inline thresholds

#define DEFAULT\_CONFIG\_InlineThreshold (35) //Default start

#define DEFAULT\_CONFIG\_AggressiveInlineThreshold (80) //Limit for aggressive inlining.

#define DEFAULT\_CONFIG\_InlineThresholdAdjustCountInLargeFunction (20)

#define DEFAULT\_CONFIG\_InlineThresholdAdjustCountInMediumSizedFunction (6)

#define DEFAULT\_CONFIG\_InlineThresholdAdjustCountInSmallFunction (10)

#define DEFAULT\_CONFIG\_ConstructorInlineThreshold (21) //Monomorphic constructor threshold

#define DEFAULT\_CONFIG\_ConstructorCallsRequiredToFinalizeCachedType (2)

#define DEFAULT\_CONFIG\_OutsideLoopInlineThreshold (16) //Threshold to inline outside loops

#define DEFAULT\_CONFIG\_LeafInlineThreshold (60) //Inlinee threshold for function which is leaf (irrespective of it has loops or not)

#define DEFAULT\_CONFIG\_LoopInlineThreshold (25) //Inlinee threshold for function with loops

#define DEFAULT\_CONFIG\_PolymorphicInlineThreshold (35) //Polymorphic inline threshold

#define DEFAULT\_CONFIG\_InlineCountMax (1200) //Max sum of bytecodes of inlinees inlined into a function (excluding built-ins)

#define DEFAULT\_CONFIG\_AggressiveInlineCountMax (8000) //Max sum of bytecodes of inlinees inlined into a function (excluding built-ins) when inlined aggressively

#define DEFAULT\_CONFIG\_MaxFuncInlineDepth (2) //Maximum number of times a function can be inlined within a top function

#define DEFAULT\_CONFIG\_MaxNumberOfInlineesWithLoop (40) //Inlinee with a loop is controlled by LoopInlineThreshold, though we don't want to inline lot of inlinees with loop, this ensures a limit.

#define DEFAULT\_CONFIG\_ConstantArgumentInlineThreshold (157) // Bytecode threshold for functions with constant arguments which are used for branching

#define DEFAULT\_CONFIG\_RecursiveInlineThreshold (2000) // Bytecode threshold recursive call at a call site

#define DEFAULT\_CONFIG\_RecursiveInlineDepthMax (8) // Maximum inline depth for recursive calls

#define DEFAULT\_CONFIG\_RecursiveInlineDepthMin (2) // Minimum inline depth for recursive calls

#define DEFAULT\_CONFIG\_CloneInlinedPolymorphicCaches (true)

#define DEFAULT\_CONFIG\_HighPrecisionDate (false)

#define DEFAULT\_CONFIG\_TDZ (true)

#define DEFAULT\_CONFIG\_ForceOldDateAPI (false)

#define DEFAULT\_CONFIG\_Loop (1)

#define DEFAULT\_CONFIG\_ForceDiagnosticsMode (false)

#define DEFAULT\_CONFIG\_EnableJitInDiagMode (true)

#define DEFAULT\_CONFIG\_EnableJitInHybridDebugging (true)

#define DEFAULT\_CONFIG\_UseFullName (true)

#define DEFAULT\_CONFIG\_EnableContinueAfterExceptionWrappersForHelpers (true)

#define DEFAULT\_CONFIG\_EnableContinueAfterExceptionWrappersForBuiltIns (true)

#define DEFAULT\_CONFIG\_EnableFunctionSourceReportForHeapEnum (true)

#define DEFAULT\_CONFIG\_LoopInterpretCount (150)

#define DEFAULT\_CONFIG\_LoopProfileIterations (25)

#define DEFAULT\_CONFIG\_JitLoopBodyHotLoopThreshold (20000)

#define DEFAULT\_CONFIG\_LoopBodySizeThresholdToDisableOpts (255)

#define DEFAULT\_CONFIG\_MaxJitThreadCount (2)

#define DEFAULT\_CONFIG\_ForceMaxJitThreadCount (false)

#ifdef RECYCLER\_PAGE\_HEAP

#define DEFAULT\_CONFIG\_PageHeap ((Js::Number) PageHeapMode::PageHeapModeOff)

#define DEFAULT\_CONFIG\_PageHeapAllocStack (false)

#define DEFAULT\_CONFIG\_PageHeapFreeStack (false)

#define DEFAULT\_CONFIG\_PageHeapBlockType ((Js::Number) PageHeapBlockTypeFilter::PageHeapBlockTypeFilterAll)

#endif

#define DEFAULT\_CONFIG\_LowMemoryCap (0xB900000) // 185 MB - based on memory cap for process on low-capacity device

#define DEFAULT\_CONFIG\_MaxCodeFill (500)

#define DEFAULT\_CONFIG\_MaxLoopsPerFunction (10)

#define DEFAULT\_CONFIG\_NopFrequency (8)

#define DEFAULT\_CONFIG\_SpeculationCap (1) // Needs to be 1 and not 0 since the compiler complains about a condition being always false

#define DEFAULT\_CONFIG\_ProfileBasedSpeculationCap (1600)

#define DEFAULT\_CONFIG\_Verbose (false)

#define DEFAULT\_CONFIG\_ForceStrictMode (false)

#define DEFAULT\_CONFIG\_EnableEvalMapCleanup (true)

#define DEFAULT\_CONFIG\_ExpirableCollectionGCCount (5) // Number of GCs during which entry point profiling occurs

#define DEFAULT\_CONFIG\_ExpirableCollectionTriggerThreshold (50) // Threshold at which Entry Point Collection is triggered

#define DEFAULT\_CONFIG\_RegexTracing (false)

#define DEFAULT\_CONFIG\_RegexProfile (false)

#define DEFAULT\_CONFIG\_RegexDebug (false)

#define DEFAULT\_CONFIG\_RegexOptimize (true)

#define DEFAULT\_CONFIG\_DynamicRegexMruListSize (16)

#define DEFAULT\_CONFIG\_GoptCleanupThreshold (25)

#define DEFAULT\_CONFIG\_AsmGoptCleanupThreshold (500)

#define DEFAULT\_CONFIG\_OptimizeForManyInstances (false)

#define DEFAULT\_CONFIG\_DeferParseThreshold (4 \* 1024) // Unit is number of characters

#define DEFAULT\_CONFIG\_ProfileBasedDeferParseThreshold (100) // Unit is number of characters

#define DEFAULT\_CONFIG\_ProfileBasedSpeculativeJit (true)

#define DEFAULT\_CONFIG\_WininetProfileCache (true)

#define DEFAULT\_CONFIG\_MinProfileCacheSize (5) // Minimum number of functions before profile is saved.

#define DEFAULT\_CONFIG\_ProfileDifferencePercent (15) // If 15% of the functions have different profile we will trigger a save.

// ES6 - BLUE

#define DEFAULT\_CONFIG\_\_\_proto\_\_ (true)

#define DEFAULT\_CONFIG\_LetConst (true)

#define DEFAULT\_CONFIG\_WeakMap (true)

#define DEFAULT\_CONFIG\_Map (true)

#define DEFAULT\_CONFIG\_Set (true)

#define DEFAULT\_CONFIG\_Intl (true)

#define DEFAULT\_CONFIG\_IntlBuiltIns (true)

// ES6 DEFAULT BEHAVIOR

#define DEFAULT\_CONFIG\_ES6 (true) // master flag to gate all P0-spec-test compliant ES6 features

//CollectGarbage is legacy IE specific global function disabled in Microsoft Edge.

#define DEFAULT\_CONFIG\_CollectGarbage (false)

// ES6 sub-feature gate - to enable-disable ES6 sub-feature when ES6 flag is enabled

#define DEFAULT\_CONFIG\_ES6Species (true)

#define DEFAULT\_CONFIG\_ES6AsyncAwait (false)

#define DEFAULT\_CONFIG\_ES6Classes (true)

#define DEFAULT\_CONFIG\_ES6DateParseFix (true)

#define DEFAULT\_CONFIG\_ES6DefaultArgs (false)

#define DEFAULT\_CONFIG\_ES6Destructuring (true)

#define DEFAULT\_CONFIG\_ES6ForLoopSemantics (true)

#define DEFAULT\_CONFIG\_ES6FunctionName (true)

#define DEFAULT\_CONFIG\_ES6FunctionNameFull (false)

#define DEFAULT\_CONFIG\_ES6Generators (true)

#define DEFAULT\_CONFIG\_ES6Iterators (true)

#define DEFAULT\_CONFIG\_ES6IsConcatSpreadable (false)

#define DEFAULT\_CONFIG\_ES6Lambda (true)

#define DEFAULT\_CONFIG\_ES6Math (true)

#define DEFAULT\_CONFIG\_ES6Object (true)

#define DEFAULT\_CONFIG\_ES6Number (true)

#define DEFAULT\_CONFIG\_ES6NumericLiterals (true)

#define DEFAULT\_CONFIG\_ES6ObjectLiterals (true)

#define DEFAULT\_CONFIG\_ES6Promise (true)

#define DEFAULT\_CONFIG\_ES6Proxy (true)

#define DEFAULT\_CONFIG\_ES6Rest (true)

#define DEFAULT\_CONFIG\_ES6Spread (true)

#define DEFAULT\_CONFIG\_ES6String (true)

#define DEFAULT\_CONFIG\_ES6StringPrototypeFixes (true)

#define DEFAULT\_CONFIG\_ES6StringTemplate (true)

#define DEFAULT\_CONFIG\_ES6PrototypeChain (false)

#define DEFAULT\_CONFIG\_ES6Symbol (true)

#define DEFAULT\_CONFIG\_ES6ToPrimitive (false)

#define DEFAULT\_CONFIG\_ES6ToLength (false)

#define DEFAULT\_CONFIG\_ES6ToStringTag (false)

#define DEFAULT\_CONFIG\_ES6TypedArrayExtensions (true)

#define DEFAULT\_CONFIG\_ES6Unicode (true)

#define DEFAULT\_CONFIG\_ES6UnicodeVerbose (true)

#define DEFAULT\_CONFIG\_ES6Unscopables (true)

#define DEFAULT\_CONFIG\_ES6WeakSet (true)

#define DEFAULT\_CONFIG\_ES6RegExSticky (true)

#define DEFAULT\_CONFIG\_ES6RegExPrototypeProperties (false)

#define DEFAULT\_CONFIG\_ES6HasInstanceOf (false)

#define DEFAULT\_CONFIG\_ArrayBufferTransfer (false)

#define DEFAULT\_CONFIG\_ES7ExponentionOperator (false)

#define DEFAULT\_CONFIG\_ES7Builtins (false)

#define DEFAULT\_CONFIG\_ES7TrailingComma (true)

#define DEFAULT\_CONFIG\_ES6Verbose (false)

#define DEFAULT\_CONFIG\_ES6All (false)

// ES6 DEFAULT BEHAVIOR

#define DEFAULT\_CONFIG\_AsyncDebugging (true)

#define DEFAULT\_CONFIG\_TraceAsyncDebugCalls (false)

#define DEFAULT\_CONFIG\_ForcePostLowerGlobOptInstrString (false)

#define DEFAULT\_CONFIG\_EnumerateSpecialPropertiesInDebugger (true)

#endif

#define DEFAULT\_CONFIG\_MaxJITFunctionBytecodeSize (120000)

#define DEFAULT\_CONFIG\_JitQueueThreshold (6)

#define DEFAULT\_CONFIG\_FullJitRequeueThreshold (25) // Minimum number of times a function needs to be executed before it is re-added to the jit queue

#define DEFAULT\_CONFIG\_MinTemplatizedJitRunCount (100) // Minimum number of times a function needs to be interpreted before it is jitted

#define DEFAULT\_CONFIG\_MinAsmJsInterpreterRunCount (10) // Minimum number of times a function needs to be Asm interpreted before it is jitted

#define DEFAULT\_CONFIG\_MinTemplatizedJitLoopRunCount (500) // Minimum number of times a function needs to be interpreted before it is jitted

#define DEFAULT\_CONFIG\_MaxTemplatizedJitRunCount (-1) // Maximum number of times a function can be TJ before it is jitted

#define DEFAULT\_CONFIG\_MaxAsmJsInterpreterRunCount (-1) // Maximum number of times a function can be Asm interpreted before it is jitted

// Note: The following defaults only apply when the NewSimpleJit is on. The defaults for when it's off are computed in

// ConfigFlagsTable::TranslateFlagConfiguration.

#define DEFAULT\_CONFIG\_AutoProfilingInterpreter0Limit (12)

#define DEFAULT\_CONFIG\_ProfilingInterpreter0Limit (4)

#define DEFAULT\_CONFIG\_AutoProfilingInterpreter1Limit (0)

#define DEFAULT\_CONFIG\_SimpleJitLimit (132)

#define DEFAULT\_CONFIG\_ProfilingInterpreter1Limit (12)

// These are used to compute the above defaults for when NewSimpleJit is off

#define DEFAULT\_CONFIG\_AutoProfilingInterpreterLimit\_OldSimpleJit (80)

#define DEFAULT\_CONFIG\_SimpleJitLimit\_OldSimpleJit (25)

#define DEFAULT\_CONFIG\_MinProfileIterations (16)

#define DEFAULT\_CONFIG\_MinProfileIterations\_OldSimpleJit (25)

#define DEFAULT\_CONFIG\_MinSimpleJitIterations (16)

#define DEFAULT\_CONFIG\_NewSimpleJit (false)

#define DEFAULT\_CONFIG\_MaxLinearIntCaseCount (3) // Maximum number of cases (in switch statement) for which instructions can be generated linearly.

#define DEFAULT\_CONFIG\_MaxSingleCharStrJumpTableRatio (2) // Maximum single char string jump table size as multiples of the actual case arm

#define DEFAULT\_CONFIG\_MaxSingleCharStrJumpTableSize (128) // Maximum single char string jump table size

#define DEFAULT\_CONFIG\_MinSwitchJumpTableSize (9) // Minimum number of case target entries in the jump table(this may also include values that are missing in the consecutive set of integer case arms)

#define DEFAULT\_CONFIG\_SwitchOptHolesThreshold (50) // Maximum percentage of holes (missing case values in a switch statement) with which a jump table can be created

#define DEFAULT\_CONFIG\_MaxLinearStringCaseCount (4) // Maximum number of String cases (in switch statement) for which instructions can be generated linearly.

#define DEFAULT\_CONFIG\_MinDeferredFuncTokenCount (20) // Minimum size in tokens of a defer-parsed function

#if DBG

#define DEFAULT\_CONFIG\_SkipFuncCountForBailOnNoProfile (0) //Initial Number of functions in a func body to be skipped from forcibly inserting BailOnNoProfile.

#endif

#define DEFAULT\_CONFIG\_BailOnNoProfileLimit 200 // The limit of bailout on no profile info before triggering a rejit

#define DEFAULT\_CONFIG\_BailOnNoProfileRejitLimit (-1) // The limit of bailout on no profile info before disable all the no profile bailouts

#define DEFAULT\_CONFIG\_RejitRatioLimit 5 // Ratio of function calls to bailouts on a single bailout record

// above which a rejit is considered

#define DEFAULT\_CONFIG\_MinBailOutsBeforeRejit 2 // Minimum number of bailouts for a single bailout record after which a rejit is considered

#define DEFAULT\_CONFIG\_RejitMaxBailOutCount 500 // Maximum number of bailouts for a single bailout record after which rejit is forced.

#define DEFAULT\_CONFIG\_Sse (-1)

#define DEFAULT\_CONFIG\_DeletedPropertyReuseThreshold (32)

#define DEFAULT\_CONFIG\_BigDictionaryTypeHandlerThreshold (0xffff)

#define DEFAULT\_CONFIG\_ForceStringKeyedSimpleDictionaryTypeHandler (false)

#define DEFAULT\_CONFIG\_TypeSnapshotEnumeration (true)

#define DEFAULT\_CONFIG\_EnumerationCompat (false)

#define DEFAULT\_CONFIG\_ConcurrentRuntime (false)

#define DEFAULT\_CONFIG\_PrimeRecycler (false)

#define DEFAULT\_CONFIG\_DisableRentalThreading (false)

#define DEFAULT\_CONFIG\_DisableDebugObject (false)

#define DEFAULT\_CONFIG\_DumpHeap (false)

#define DEFAULT\_CONFIG\_PerfHintLevel (1)

#define DEFAULT\_CONFIG\_FailFastIfDisconnectedDelegate (false)

#define DEFAULT\_CONFIG\_IsolatePrototypes (true)

#define DEFAULT\_CONFIG\_ChangeTypeOnProto (true)

#define DEFAULT\_CONFIG\_FixPropsOnPathTypes (true)

#define DEFAULT\_CONFIG\_BailoutTraceFilter (-1)

#define DEFAULT\_CONFIG\_TempMin (0)

#define DEFAULT\_CONFIG\_TempMax (MAXINT)

#define DEFAULT\_CONFIG\_LibraryStackFrame (true)

#define DEFAULT\_CONFIG\_LibraryStackFrameDebugger (false)

#define DEFAULT\_CONFIG\_FuncObjectInlineCacheThreshold (2) // Maximum number of inline caches a function body may have to allow for inline caches to be allocated on the function object.

#define DEFAULT\_CONFIG\_ShareInlineCaches (true)

#define DEFAULT\_CONFIG\_InlineCacheInvalidationListCompactionThreshold (4)

#define DEFAULT\_CONFIG\_InMemoryTrace (false)

#define DEFAULT\_CONFIG\_InMemoryTraceBufferSize (1024)

#define DEFAULT\_CONFIG\_RichTraceFormat (false)

#define DEFAULT\_CONFIG\_TraceWithStack (false)

#define DEFAULT\_CONFIG\_InjectPartiallyInitializedInterpreterFrameError (0)

#define DEFAULT\_CONFIG\_InjectPartiallyInitializedInterpreterFrameErrorType (0)

#define DEFAULT\_CONFIG\_InvalidateSolutionContextsForGetStructure (true)

#define DEFAULT\_CONFIG\_DeferLoadingAvailableSource (false)

#ifdef ENABLE\_PROJECTION

#define DEFAULT\_CONFIG\_NoWinRTFastSig (false)

#define DEFAULT\_CONFIG\_TargetWinRTVersion (1)

#define DEFAULT\_CONFIG\_WinRTDelegateInterfaces (false)

#define DEFAULT\_CONFIG\_WinRTAdaptiveApps (true)

#endif

#define DEFAULT\_CONFIG\_RecyclerForceMarkInterior (false)

#define DEFAULT\_CONFIG\_MemProtectHeap (false)

#define DEFAULT\_CONFIG\_InduceCodeGenFailure (30) // When -InduceCodeGenFailure is passed in, 30% of JIT allocations will fail

#define DEFAULT\_CONFIG\_SkipSplitWhenResultIgnored (false)

#define DEFAULT\_CONFIG\_MinMemOpCount (16U)

#if ENABLE\_COPYONACCESS\_ARRAY

#define DEFAULT\_CONFIG\_MaxCopyOnAccessArrayLength (32U)

#define DEFAULT\_CONFIG\_MinCopyOnAccessArrayLength (5U)

#define DEFAULT\_CONFIG\_CopyOnAccessArraySegmentCacheSize (16U)

#endif

#if defined(\_M\_IX86) || defined(\_M\_X64)

#define DEFAULT\_CONFIG\_LoopAlignNopLimit (6)

#endif

#define TraceLevel\_Error (1)

#define TraceLevel\_Warning (2)

#define TraceLevel\_Info (3)

#define TEMP\_ENABLE\_FLAG\_FOR\_APPX\_BETA\_ONLY 1

#define INMEMORY\_CACHE\_MAX\_URL (5) // This is the max number of URLs that the in-memory profile cache can hold.

#define INMEMORY\_CACHE\_MAX\_PROFILE\_MANAGER (50) // This is the max number of dynamic scripts that the in-memory profile cache can have

#ifdef SUPPORT\_INTRUSIVE\_TESTTRACES

#define INTRUSIVE\_TESTTRACE\_PolymorphicInlineCache (1)

#endif

//

//FLAG(type, name, description, defaultValue)

//

// types:

// String

// Phases

// Number

// Boolean

//

// If the default value is not required it should be left empty

// For Phases, there is no default value. it should always be empty

// Default values for stings must be prefixed with 'L'. See AsmDumpMode

// Scroll till the extreme right to see the default values

#if defined(FLAG) || defined(FLAG\_REGOVR\_EXP)

#ifndef FLAG

#define FLAG(...)

#endif

#ifndef FLAG\_REGOVR\_ASMJS

#define FLAG\_REGOVR\_ASMJS FLAG

#endif

#ifndef FLAG\_REGOVR\_EXP

#define FLAG\_REGOVR\_EXP FLAG

#endif

// NON-RELEASE FLAGS

#ifdef ENABLE\_DEBUG\_CONFIG\_OPTIONS

// Regular FLAG

#define FLAGNR(Type, Name, String, Default) FLAG(Type, Name, String, Default, NoParent, FALSE)

// Regular flag with acronym

#ifndef FLAGNRA

#define FLAGNRA(Type, Name, Acronym, String, Default) \

FLAGNR(Type, Name, String, Default) \

FLAGNR(Type, Acronym, String, Default)

#endif

// Child FLAG with PARENT FLAG

#define FLAGPNR(Type, ParentName, Name, String, Default) FLAG(Type, Name, String, Default, ParentName, FALSE)

// Regular FLAG with callback function

#define FLAGNRC(Type, Name, String, Default) FLAG(Type, Name, String, Default, NoParent, TRUE)

#else

#define FLAGNR(Type, Name, String, Default)

#ifdef FLAGNRA

#undef FLAGNRA

#endif

#define FLAGNRA(Type, Name, Acronym, String, Default)

#define FLAGPNR(Type, ParentName, Name, String, Default)

#define FLAGNRC(Type, Name, String, Default)

#endif

// RELEASE FLAGS

#define FLAGPR(Type, ParentName, Name, String, Default) FLAG(Type, Name, String, Default, ParentName, FALSE)

#define FLAGR(Type, Name, String, Default) FLAG(Type, Name, String, Default, NoParent, FALSE)

// RELEASE FLAGS WITH REGISTRY OVERRIDE

#define FLAGPR\_REGOVR\_ASMJS(Type, ParentName, Name, String, Default) FLAG\_REGOVR\_ASMJS(Type, Name, String, Default, ParentName, FALSE)

#define FLAGPR\_REGOVR\_EXP(Type, ParentName, Name, String, Default) FLAG\_REGOVR\_EXP(Type, Name, String, Default, ParentName, FALSE)

// Release flag with non-release acronym

#ifndef FLAGRA

#define FLAGRA(Type, Name, Acronym, String, Default) \

FLAGR(Type, Name, String, Default) \

FLAGNR(Type, Acronym, String, Default)

#endif

// Please keep this list alphabetically sorted

#if DBG

FLAGNR(Boolean, ArrayValidate , "Validate each array for valid elements (default: false)", false)

FLAGNR(Boolean, MemOpMissingValueValidate, "Validate Missing Value Tracking on memset/memcopy", false)

#endif

#ifdef ARENA\_MEMORY\_VERIFY

FLAGNR(Boolean, ArenaNoFreeList , "Do not free list in arena", false)

FLAGNR(Boolean, ArenaNoPageReuse , "Do not reuse page in arena", false)

FLAGNR(Boolean, ArenaUseHeapAlloc , "Arena use heap to allocate memory instead of page allocator", false)

#endif

FLAGNR(Boolean, ValidateInlineStack, "Does a stack walk on helper calls to validate inline stack is correctly restored", false)

FLAGNR(Boolean, AsmDiff , "Dump the IR without memory locations and varying parameters.", false)

FLAGNR(String, AsmDumpMode , "Dump the final assembly to a file without memory locations and varying parameters\n\t\t\t\t\tThe 'filename' is the file where the assembly will be dumped. Dump to console if no file is specified", nullptr)

FLAGR (Boolean, Asmjs , "Enable Asmjs", DEFAULT\_CONFIG\_ASMJS)

FLAGNR(Boolean, AsmJsStopOnError , "Stop execution on any AsmJs validation errors", DEFAULT\_CONFIG\_AsmJsStopOnError)

FLAGNR(Boolean, AsmJsEdge , "Enable asm.js features which may have backward incompatible changes or not validate on old demos", DEFAULT\_CONFIG\_AsmJsEdge)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, Simdjs, "Enable Simdjs", DEFAULT\_CONFIG\_SIMDJS)

FLAGR(Boolean, Simd128TypeSpec, "Enable type-specialization of Simd128 symbols", false)

FLAGNR(Boolean, AssertBreak , "Debug break on assert", false)

FLAGNR(Boolean, AssertPopUp , "Pop up asserts (default: false)", false)

FLAGNR(Boolean, AssertIgnore , "Ignores asserts if set", false)

FLAGNR(Boolean, AsyncDebugging, "Enable async debugging feature (default: false)", DEFAULT\_CONFIG\_AsyncDebugging)

FLAGNR(Number, BailOnNoProfileLimit, "The limit of bailout on no profile info before triggering a rejit", DEFAULT\_CONFIG\_BailOnNoProfileLimit)

FLAGNR(Number, BailOnNoProfileRejitLimit, "The limit of bailout on no profile info before we disable the bailouts", DEFAULT\_CONFIG\_BailOnNoProfileRejitLimit)

FLAGNR(String, DumpOnCrash , "generate heap dump on asserts or unhandled exception if set", nullptr)

FLAGNR(String, FullMemoryDump , "Will perform a full memory dump when -DumpOnCrash is supplied.", nullptr)

#ifdef BAILOUT\_INJECTION

FLAGR (NumberPairSet, BailOut , "Source location to insert BailOut", )

FLAGNR(Boolean, BailOutAtEveryLine , "Inserts BailOut at every line of source (default: false)", false)

FLAGNR(Boolean, BailOutAtEveryByteCode, "Inserts BailOut at every Byte code (default: false)", false)

FLAGNR(Boolean, BailOutAtEveryImplicitCall, "Force generating implicit call bailout even when we don't need it", false)

FLAGR (NumberSet, BailOutByteCode , "Byte code location to insert BailOut. Use with -prejit only", )

#endif

FLAGNR(Boolean, Benchmark , "Disable security code which introduce variability in benchmarks", false)

FLAGR (Boolean, BgJit , "Background JIT. Disable to force heuristic-based foreground JITting. (default: true)", true)

FLAGNR(Number, BgJitDelay , "Delay to wait for speculative jitting before starting script execution", DEFAULT\_CONFIG\_BgJitDelay)

FLAGNR(Number, BgJitDelayFgBuffer , "When speculatively jitting in the foreground thread, do so for (BgJitDelay - BgJitDelayBuffer) milliseconds", DEFAULT\_CONFIG\_BgJitDelayFgBuffer)

FLAGNR(Number, BgJitPendingFuncCap , "Disable delay if pending function count larger then cap", DEFAULT\_CONFIG\_BgJitPendingFuncCap)

FLAGNR(Boolean, CreateFunctionProxy , "Create function proxies instead of full function bodies", DEFAULT\_CONFIG\_CreateFunctionProxy)

FLAGNR(Boolean, HybridFgJit , "When background JIT is enabled, enable jitting in the foreground based on heuristics. This flag is only effective when OptimizeForManyInstances is disabled (UI threads).", DEFAULT\_CONFIG\_HybridFgJit)

FLAGNR(Number, HybridFgJitBgQueueLengthThreshold, "The background job queue length must exceed this threshold to consider jitting in the foreground", DEFAULT\_CONFIG\_HybridFgJitBgQueueLengthThreshold)

FLAGNR(Boolean, BytecodeHist , "Provide a histogram of the bytecodes run by the script. (NoNative required).", false)

FLAGNR(Boolean, CurrentSourceInfo , "Enable IASD get current script source info", DEFAULT\_CONFIG\_CurrentSourceInfo)

FLAGNR(Boolean, CFGLog , "Log CFG checks", false)

FLAGNR(Boolean, CheckAlignment , "Insert checks in the native code to verify 8-byte alignment of stack", false)

FLAGNR(Boolean, CheckEmitBufferPermissions, "Check JIT code buffers at commit and decommit time to ensure no PAGE\_EXECUTE\_READWRITE pages.", false)

#ifdef CHECK\_MEMORY\_LEAK

FLAGR (Boolean, CheckMemoryLeak , "Check for heap memory leak", false)

FLAGR (String, DumpOnLeak , "Create a dump on failed memory leak check", nullptr)

#endif

FLAGNR(Boolean, CheckOpHelpers , "Verify opHelper labels in the JIT are set properly", false)

FLAGNR(Boolean, CloneInlinedPolymorphicCaches, "Clones polymorphic inline caches in inlined functions", DEFAULT\_CONFIG\_CloneInlinedPolymorphicCaches)

FLAGNR(Boolean, ConcurrentRuntime , "Enable Concurrent GC and background JIT when creating runtime", DEFAULT\_CONFIG\_ConcurrentRuntime)

FLAGNR(Boolean, Console , "Create console window in GUI app", false)

FLAGNR(Boolean, ConsoleExitPause , "Pause on exit when a console window is created in GUI app", false)

FLAGNR(Number, ConstructorInlineThreshold , "Maximum size in bytecodes of an constructor inline candidate with monomorphic field access", DEFAULT\_CONFIG\_ConstructorInlineThreshold)

FLAGNR(Number, ConstructorCallsRequiredToFinalizeCachedType, "Number of calls to a constructor required before the type cached in the constructor cache is finalized", DEFAULT\_CONFIG\_ConstructorCallsRequiredToFinalizeCachedType)

#ifdef SECURITY\_TESTING

FLAGNR(Boolean, CrashOnException , "Removes the top-level exception handler, allowing jc.exe to crash on an unhandled exception. No effect on IE. (default: false)", false)

#endif

FLAGNR(Boolean, Debug , "Disable phases (layout, security code, etc) which makes JIT output harder to debug", false)

FLAGNR(NumberSet, DebugBreak , "Index of the function where you want to break", )

FLAGNR(Boolean, DebugWindow , "Send console output to debugger window", false)

FLAGNR(Boolean, DeferNested , "Enable deferred parsing of nested function", DEFAULT\_CONFIG\_DeferNested)

FLAGNR(Boolean, DeferTopLevelTillFirstCall , "Enable tracking of deferred top level functions in a script file, until the first function of the script context is parsed.", DEFAULT\_CONFIG\_DeferTopLevelTillFirstCall)

FLAGNR(Number, DeferParse , "Minimum size of defer-parsed script (non-zero only: use /nodeferparse do disable", 0)

FLAGNR(Boolean, DefineGetterSetter , "Enable IE11 \_\_defineGetter\_\_ and \_\_defineSetter\_\_ methods", DEFAULT\_CONFIG\_DefineGetterSetter)

FLAGNR(Boolean, DirectCallTelemetryStats, "Enables logging stats for direct call telemetry", DEFAULT\_CONFIG\_DirectCallTelemetryStats)

FLAGNR(Boolean, DisableArrayBTree , "Disable creation of BTree for Arrays", false)

FLAGNR(Boolean, DisableRentalThreading, "Disable rental threading when creating runtime", DEFAULT\_CONFIG\_DisableRentalThreading)

FLAGNR(Boolean, DisableVTuneSourceLineInfo, "Disable VTune Source line info for Dynamic JITted code", false)

FLAGNR(Boolean, DisplayMemStats, "Display memory usage statistics", false)

FLAGNR(Phases, Dump , "What All to dump", )

#ifdef DUMP\_FRAGMENTATION\_STATS

FLAGR (Boolean, DumpFragmentationStats, "Dump bucket state after every GC", false)

#endif

FLAGNR(Boolean, DumpIRAddresses, "Print addresses in IR dumps", false)

FLAGNR(Boolean, DumpLineNoInColor, "Print the source code in high intensity color for better readability", false)

#ifdef RECYCLER\_DUMP\_OBJECT\_GRAPH

FLAGR (Boolean, DumpObjectGraphOnExit , "Dump object graph on recycler destructor", false)

FLAGR (Boolean, DumpObjectGraphOnCollect, "Dump object graph on recycler destructor", false)

#endif

FLAGNR(Boolean, DumpEvalStringOnRemoval, "Dumps an eval string when its being removed from the eval map", false)

FLAGNR(Boolean, DumpObjectGraphOnEnum, "Dump object graph on recycler heap enumeration", false)

#ifdef DYNAMIC\_PROFILE\_STORAGE

FLAGNRA(String, DynamicProfileCache , Dpc, "File to cache dynamic profile information", nullptr)

FLAGNR(String, DynamicProfileCacheDir, "Directory to cache dynamic profile information", nullptr)

FLAGNRA(String, DynamicProfileInput , Dpi, "Read only file containing dynamic profile information", nullptr)

#endif

#ifdef EDIT\_AND\_CONTINUE

FLAGNR(Boolean, EditTest , "Enable edit and continue test tools", false)

#endif

FLAGNR(Boolean, WininetProfileCache, "Use the WININET cache to save the profile information", DEFAULT\_CONFIG\_WininetProfileCache)

FLAGNR(Boolean, NoDynamicProfileInMemoryCache, "Enable in-memory cache for dynamic sources", false)

FLAGNR(Boolean, ProfileBasedSpeculativeJit, "Enable dynamic profile based speculative JIT", DEFAULT\_CONFIG\_ProfileBasedSpeculativeJit)

FLAGNR(Number, ProfileBasedSpeculationCap, "In the presence of dynamic profile speculative JIT is capped to this many bytecode instructions", DEFAULT\_CONFIG\_ProfileBasedSpeculationCap)

#ifdef DYNAMIC\_PROFILE\_MUTATOR

FLAGNR(String, DynamicProfileMutatorDll , "Path of the mutator DLL", L"DynamicProfileMutatorImpl.dll")

FLAGNR(String, DynamicProfileMutator , "Type of local, temp, return, param, loop implicit flag and implicit flag. \n\t\t\t\t\ti.e local=LikelyArray\_NoMissingValues\_NonInts\_NonFloats;temp=Int8Array;param=LikelyNumber;return=LikelyString;loopimplicitflag=ImplicitCall\_ToPrimitive;implicitflag=ImplicitCall\_None\n\t\t\t\t\tor pass DynamicProfileMutator:random\n\t\t\t\t\tSee DynamicProfileInfo.h for enum values", nullptr)

#endif

FLAGNR(Boolean, ExecuteByteCodeBufferReturnsInvalidByteCode, "Serialized byte code execution always returns SCRIPT\_E\_INVALID\_BYTECODE", false)

FLAGR(Number, ExpirableCollectionGCCount, "Number of GCs during which Expirable object profiling occurs", DEFAULT\_CONFIG\_ExpirableCollectionGCCount)

FLAGR (Number, ExpirableCollectionTriggerThreshold, "Threshold at which Expirable Object Collection is triggered (In Percentage)", DEFAULT\_CONFIG\_ExpirableCollectionTriggerThreshold)

FLAGR(Boolean, SkipSplitOnNoResult, "If the result of Regex split isn't used, skip executing the regex. (Perf optimization)", DEFAULT\_CONFIG\_SkipSplitWhenResultIgnored)

#ifdef TEST\_ETW\_EVENTS

FLAGNR(String, TestEtwDll , "Path of the TestEtwEventSink DLL", nullptr)

#endif

// ES6 (BLUE-shipped) features/flags

FLAGNR(Boolean, \_\_proto\_\_ , "\_\_proto\_\_ support", DEFAULT\_CONFIG\_\_\_proto\_\_)

FLAGNR(Boolean, LetConst , "Enable support of 'let' and 'const' keyword", DEFAULT\_CONFIG\_LetConst)

FLAGNR(Boolean, Map , "Enable ES6 Map feature", DEFAULT\_CONFIG\_Map)

FLAGNR(Boolean, Set , "Enable ES6 Set feature", DEFAULT\_CONFIG\_Set)

FLAGNR(Boolean, WeakMap , "Enable ES6 WeakMap feature", DEFAULT\_CONFIG\_WeakMap)

FLAGNR(Boolean, CollectGarbage , "Enable CollectGarbage API", DEFAULT\_CONFIG\_CollectGarbage)

FLAGR (Boolean, Intl , "Intl object support", DEFAULT\_CONFIG\_Intl)

FLAGNR(Boolean, IntlBuiltIns , "Intl built-in function support", DEFAULT\_CONFIG\_IntlBuiltIns)

// ES6 (BLUE+1) features/flags

// Master ES6 flag to enable STABLE ES6 features/flags

FLAGR(Boolean, ES6 , "Enable ES6 stable features", DEFAULT\_CONFIG\_ES6)

// Master ES6 flag to enable ALL sub ES6 features/flags

FLAGNRC(Boolean, ES6All , "Enable all ES6 features, both stable and unstable", DEFAULT\_CONFIG\_ES6All)

// Master ES6 flag to enable Threshold ES6 features/flags

FLAGNRC(Boolean, ES6Experimental , "Enable all experimental features", DEFAULT\_CONFIG\_ES6All)

// Per ES6 feature/flag

FLAGPR (Boolean, ES6, ES6Species , "Enable ES6 '@@species' properties and built-in behaviors" , DEFAULT\_CONFIG\_ES6Species)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES7AsyncAwait , "Enable ES7 'async' and 'await' keywords" , DEFAULT\_CONFIG\_ES6AsyncAwait)

FLAGPR (Boolean, ES6, ES6Classes , "Enable ES6 'class' and 'extends' keywords" , DEFAULT\_CONFIG\_ES6Classes)

FLAGPR (Boolean, ES6, ES6DateParseFix , "Enable ES6 Date.parse fixes" , DEFAULT\_CONFIG\_ES6DateParseFix)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES6DefaultArgs , "Enable ES6 Default Arguments" , DEFAULT\_CONFIG\_ES6DefaultArgs)

FLAGPR (Boolean, ES6, ES6Destructuring , "Enable ES6 Destructuring" , DEFAULT\_CONFIG\_ES6Destructuring)

FLAGPR (Boolean, ES6, ES6ForLoopSemantics , "Enable ES6 for loop per iteration bindings" , DEFAULT\_CONFIG\_ES6ForLoopSemantics)

FLAGPR (Boolean, ES6, ES6FunctionName , "Enable ES6 function.name" , DEFAULT\_CONFIG\_ES6FunctionName)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES6FunctionNameFull , "Enable ES6 Full function.name" , DEFAULT\_CONFIG\_ES6FunctionNameFull)

FLAGPR (Boolean, ES6, ES6Generators , "Enable ES6 generators" , DEFAULT\_CONFIG\_ES6Generators)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES7ExponentiationOperator, "Enable ES7 exponentiation operator (\*\*)" , DEFAULT\_CONFIG\_ES7ExponentionOperator)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES7Builtins , "Enable ES7 built-ins" , DEFAULT\_CONFIG\_ES7Builtins)

FLAGPR (Boolean, ES6, ES7TrailingComma , "Enable ES7 trailing comma in function" , DEFAULT\_CONFIG\_ES7TrailingComma)

FLAGPR (Boolean, ES6, ES6Iterators , "Enable ES6 iterators" , DEFAULT\_CONFIG\_ES6Iterators)

FLAGPR (Boolean, ES6, ES6IsConcatSpreadable , "Enable ES6 isConcatSpreadable Symbol" , DEFAULT\_CONFIG\_ES6IsConcatSpreadable)

FLAGPR (Boolean, ES6, ES6Lambda , "Enable ES6 lambdas" , DEFAULT\_CONFIG\_ES6Lambda)

FLAGPR (Boolean, ES6, ES6Math , "Enable ES6 Math extensions" , DEFAULT\_CONFIG\_ES6Math)

FLAGPR (Boolean, ES6, ES6Object , "Enable ES6 Object extensions" , DEFAULT\_CONFIG\_ES6Object)

FLAGPR (Boolean, ES6, ES6Number , "Enable ES6 Number extensions" , DEFAULT\_CONFIG\_ES6Number)

FLAGPR (Boolean, ES6, ES6NumericLiterals , "Enable ES6 NumericLiterals" , DEFAULT\_CONFIG\_ES6NumericLiterals)

FLAGPR (Boolean, ES6, ES6ObjectLiterals , "Enable ES6 Object literal extensions" , DEFAULT\_CONFIG\_ES6ObjectLiterals)

FLAGPR (Boolean, ES6, ES6Promise , "Enable ES6 Promise feature" , DEFAULT\_CONFIG\_ES6Promise)

FLAGPR (Boolean, ES6, ES6Proxy , "Enable ES6 Proxy feature" , DEFAULT\_CONFIG\_ES6Proxy)

FLAGPR (Boolean, ES6, ES6Rest , "Enable ES6 Rest parameters" , DEFAULT\_CONFIG\_ES6Rest)

FLAGPR (Boolean, ES6, ES6Spread , "Enable ES6 Spread support" , DEFAULT\_CONFIG\_ES6Spread)

FLAGPR (Boolean, ES6, ES6String , "Enable ES6 String extensions" , DEFAULT\_CONFIG\_ES6String)

FLAGPR (Boolean, ES6, ES6StringPrototypeFixes, "Enable ES6 String.prototype fixes" , DEFAULT\_CONFIG\_ES6StringPrototypeFixes)

FLAGPR (Boolean, ES6, ES6StringTemplate , "Enable ES6 string template feature" , DEFAULT\_CONFIG\_ES6StringTemplate)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES6PrototypeChain , "Enable ES6 prototypes (Example: Date prototype is object)", DEFAULT\_CONFIG\_ES6PrototypeChain)

FLAGPR (Boolean, ES6, ES6Symbol , "Enable ES6 Symbol feature" , DEFAULT\_CONFIG\_ES6Symbol)

FLAGPR (Boolean, ES6, ES6ToPrimitive , "Enable ES6 ToPrimitve symbol" , DEFAULT\_CONFIG\_ES6ToPrimitive)

FLAGPR (Boolean, ES6, ES6ToLength , "Enable ES6 ToLength fixes" , DEFAULT\_CONFIG\_ES6ToLength)

FLAGPR (Boolean, ES6, ES6ToStringTag , "Enable ES6 ToStringTag symbol" , DEFAULT\_CONFIG\_ES6ToStringTag)

FLAGPR (Boolean, ES6, ES6TypedArrayExtensions, "Enable ES6 TypedArray extensions" , DEFAULT\_CONFIG\_ES6TypedArrayExtensions)

FLAGPR (Boolean, ES6, ES6Unicode , "Enable ES6 Unicode 6.0 extensions" , DEFAULT\_CONFIG\_ES6Unicode)

FLAGPR (Boolean, ES6, ES6UnicodeVerbose , "Enable ES6 Unicode 6.0 verbose failure output" , DEFAULT\_CONFIG\_ES6UnicodeVerbose)

FLAGPR (Boolean, ES6, ES6Unscopables , "Enable ES6 With Statement Unscopables" , DEFAULT\_CONFIG\_ES6Unscopables)

FLAGPR (Boolean, ES6, ES6WeakSet , "Enable ES6 WeakSet" , DEFAULT\_CONFIG\_ES6WeakSet)

FLAGPR (Boolean, ES6, ES6RegExSticky , "Enable ES6 RegEx sticky flag" , DEFAULT\_CONFIG\_ES6RegExSticky)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ES6RegExPrototypeProperties, "Enable ES6 properties on the RegEx prototype" , DEFAULT\_CONFIG\_ES6RegExPrototypeProperties)

FLAGPR (Boolean, ES6, ES6HasInstance , "Enable ES6 @@hasInstance symbol" , DEFAULT\_CONFIG\_ES6HasInstanceOf)

FLAGPR (Boolean, ES6, ES6Verbose , "Enable ES6 verbose trace" , DEFAULT\_CONFIG\_ES6Verbose)

FLAGPR\_REGOVR\_EXP(Boolean, ES6, ArrayBufferTransfer , "Enable ArrayBuffer.transfer" , DEFAULT\_CONFIG\_ArrayBufferTransfer)

// /ES6 (BLUE+1) features/flags

#ifdef ENABLE\_PROJECTION

FLAGNR(Boolean, WinRTDelegateInterfaces , "Treat WinRT Delegates as Interfaces when determining their resolvability.", DEFAULT\_CONFIG\_WinRTDelegateInterfaces)

FLAGR(Boolean, WinRTAdaptiveApps , "Enable the adaptive apps feature, allowing for variable projection." , DEFAULT\_CONFIG\_WinRTAdaptiveApps)

#endif

// This flag to be removed once JITing generator functions is stable

FLAGNR(Boolean, JitES6Generators , "Enable JITing of ES6 generators", false)

FLAGNR(Number, FastPathCap , "Cap in source code size for enabling fast-paths", DEFAULT\_CONFIG\_FastPathCap)

FLAGNR(Boolean, FastLineColumnCalculation, "Enable fast calculation of line/column numbers from the source.", DEFAULT\_CONFIG\_FastLineColumnCalculation)

FLAGR (String, Filename , "Jscript source file", nullptr)

FLAGNR(Boolean, FreeRejittedCode , "Free rejitted code", true)

FLAGNR(Boolean, ForceGuardPages , "Force the addition of guard pages", false)

FLAGNR(Boolean, PrintGuardPageBounds , "Prints the bounds of a guard page", false)

FLAGNR(Boolean, ForceLegacyEngine , "Force a jscrip9 dll load", false)

FLAGNR(Phases, Force , "Force certain phase to run ignoring heuristics", )

FLAGNR(Phases, Stress , "Stress certain phases by making them kick in even if they normally would not.", )

FLAGNR(Boolean, ForceArrayBTree , "Force enable creation of BTree for Arrays", false)

FLAGNR(Boolean, ForceCleanPropertyOnCollect, "Force cleaning of property on collection", DEFAULT\_CONFIG\_ForceCleanPropertyOnCollect)

FLAGNR(Boolean, ForceCleanCacheOnCollect, "Force cleaning of dynamic caches on collection", DEFAULT\_CONFIG\_ForceCleanCacheOnCollect)

FLAGNR(Boolean, ForceGCAfterJSONParse, "Force GC to happen after JSON parsing", DEFAULT\_CONFIG\_ForceGCAfterJSONParse)

FLAGNR(Boolean, ForceDecommitOnCollect, "Force decommit collect", DEFAULT\_CONFIG\_ForceDecommitOnCollect)

FLAGNR(Boolean, ForceDeferParse , "Defer parsing of all function bodies", DEFAULT\_CONFIG\_ForceDeferParse)

FLAGNR(Boolean, ForceDiagnosticsMode , "Enable diagnostics mode and debug interpreter loop", false)

FLAGNR(Boolean, ForceGetWriteWatchOOM , "Force GetWriteWatch to go into OOM codepath in HeapBlockMap rescan", false)

FLAGNR(Boolean, DumpDbgControllerBytecode, "Dump dbgcontroller.js bytecode when -dump:bytecode is specified (default false)", false)

FLAGNR(Boolean, ForcePostLowerGlobOptInstrString, "Force tracking of globopt instr string post lower", DEFAULT\_CONFIG\_ForcePostLowerGlobOptInstrString)

FLAGNR(Boolean, EnumerateSpecialPropertiesInDebugger, "Enable enumeration of special debug properties", DEFAULT\_CONFIG\_EnumerateSpecialPropertiesInDebugger)

FLAGNR(Boolean, EnableJitInDiagMode , "Enable Fast F12 (only applicable with ForceDiagnosticsMode or while under debugger)", DEFAULT\_CONFIG\_EnableJitInDiagMode)

FLAGR (Boolean, EnableJitInHybridDebugging, "Enable Fast Debugging for Hybrid Debugging. Node: to turn this ON in full, EnableJitInDiagMode must be ON as well.", DEFAULT\_CONFIG\_EnableJitInHybridDebugging)

FLAGNR(Boolean, EnableContinueAfterExceptionWrappersForHelpers, "Enable wrapper over helper methods in debugger, Fast F12 only", DEFAULT\_CONFIG\_EnableContinueAfterExceptionWrappersForHelpers)

FLAGNR(Boolean, EnableContinueAfterExceptionWrappersForBuiltIns, "Enable wrapper over library calls in debugger, Fast F12 only", DEFAULT\_CONFIG\_EnableContinueAfterExceptionWrappersForBuiltIns)

FLAGNR(Boolean, EnableFunctionSourceReportForHeapEnum, "During HeapEnum, whether to report function source info (url/row/col)", DEFAULT\_CONFIG\_EnableFunctionSourceReportForHeapEnum)

FLAGNR(Number, ForceFragmentAddressSpace , "Fragment the address space", 128 \* 1024 \* 1024)

FLAGNR(Number, ForceOOMOnEBCommit, "Force CommitBuffer to return OOM", 0)

FLAGR (Boolean, ForceDynamicProfile , "Force to always generate profiling byte code", DEFAULT\_CONFIG\_ForceDynamicProfile)

FLAGNR(Boolean, ForceES5Array , "Force using ES5Array", DEFAULT\_CONFIG\_ForceES5Array)

FLAGNR(Boolean, ForceAsmJsLinkFail , "Force asm.js link time validation to fail", DEFAULT\_CONFIG\_ForceAsmJsLinkFail)

FLAGNR(Boolean, ForceExpireOnNonCacheCollect, "Allow expiration collect outside of cache collection cleanups", DEFAULT\_CONFIG\_ForceExpireOnNonCacheCollect)

FLAGNR(Boolean, ForceFastPath , "Force fast-paths in native codegen", DEFAULT\_CONFIG\_ForceFastPath)

FLAGNR(Boolean, ForceFloatPref , "Force float preferencing (JIT only)", false)

FLAGNR(Boolean, ForceJITLoopBody , "Force jit loop body only", DEFAULT\_CONFIG\_ForceJITLoopBody)

FLAGNR(Boolean, DumpCommentsFromReferencedFiles, "Allow printing comments of comment-table of the referenced file as well (use with -trace:CommentTable)", DEFAULT\_CONFIG\_DumpCommentsFromReferencedFiles)

#ifdef \_M\_ARM

FLAGNR(Boolean, ForceLocalsPtr , "Force use of alternative locals pointer (JIT only)", false)

#endif

FLAGNR(Boolean, DeferLoadingAvailableSource, "Treat available source code as a dummy defer-mappable object to go through that code path.", DEFAULT\_CONFIG\_DeferLoadingAvailableSource)

FLAGR (Boolean, ForceNative , "Force JIT everything that is called before running it, ignoring limits", false)

FLAGNR(Boolean, ForceSerialized , "Always serialize and deserialize byte codes before execution", DEFAULT\_CONFIG\_ForceSerialized)

FLAGNR(Number, ForceSerializedBytecodeMajorVersion, "Force the byte code serializer to write this major version number", 0)

FLAGNR(Number, ForceSerializedBytecodeVersionSchema, "Force the byte code serializer to write this kind of version. Decimal 10 is engineering, 20 is release mode, and 0 means use the default setting.", 0)

FLAGNR(Boolean, ForceStrictMode, "Force strict mode checks on all functions", false)

FLAGNR(Boolean, ForceUndoDefer , "Defer parsing of all function bodies, but undo deferral", false)

FLAGNR(Boolean, ForceBlockingConcurrentCollect, "Force doing in-thread GC on concurrent thread- this will skip doing concurrent collect", false)

FLAGNR(Boolean, FreTestDiagMode, "Enabled collection of diagnostic information on fretest builds", false)

#ifdef BYTECODE\_TESTING

FLAGNR(Number, ByteCodeBranchLimit, "Short branch limit before we use the branch island", 128)

FLAGNR(Boolean, MediumByteCodeLayout , "Always use medium layout for bytecodes", false)

FLAGNR(Boolean, LargeByteCodeLayout , "Always use large layout for bytecodes", false)

#endif

#ifdef FAULT\_INJECTION

FLAGNR(Number, FaultInjection , "FaultInjectMode - 0 (count only), 1 (count equal), 2 (count at or above), 3 (stackhashing)",-1)

FLAGNR(Number, FaultInjectionCount , "Injects an out of memory at the specified allocation", -1)

FLAGNR(String, FaultInjectionType , "FaultType (flag values) - 1 (Throw), 2 (NoThrow), 4 (MarkThrow), 8 (MarkNoThrow), FFFFFFFF (All)", nullptr)

FLAGNR(String, FaultInjectionFilter , "A string to restrict the fault injection, the string can be like ArenaAllocator name", nullptr)

FLAGNR(Number, FaultInjectionAllocSize, "Do fault injection only this size", -1)

FLAGNR(String, FaultInjectionStackFile , "Stacks to match, default: stack.txt in current directory", L"stack.txt")

FLAGNR(Number, FaultInjectionStackLineCount , "Count of lines in the stack file used for matching", -1)

FLAGNR(String, FaultInjectionStackHash, "Match stacks hash on Chakra frames to inject the fault, hex string", L"0")

FLAGNR(Number, FaultInjectionScriptContextToTerminateCount, "Script context# COUNT % (Number of script contexts) to terminate", 1)

#endif

FLAGNR(Number, InduceCodeGenFailure, "Probability of a codegen job failing.", DEFAULT\_CONFIG\_InduceCodeGenFailure)

FLAGNR(Number, InduceCodeGenFailureSeed, "Seed used while calculating codegen failure probability", 0)

FLAGNR(Number, InjectPartiallyInitializedInterpreterFrameError, "The number of interpreter stack frame (with 1 being bottom-most) to inject error before the frame is initialized.", DEFAULT\_CONFIG\_InjectPartiallyInitializedInterpreterFrameError)

FLAGNR(Number, InjectPartiallyInitializedInterpreterFrameErrorType, "Type of error to inject: 0 - debug break, 1 - exception.", DEFAULT\_CONFIG\_InjectPartiallyInitializedInterpreterFrameErrorType)

FLAGNR(Boolean, GenerateByteCodeBufferReturnsCantGenerate, "Serialized byte code generation always returns SCRIPT\_E\_CANT\_GENERATE", false)

FLAGNR(Number, GoptCleanupThreshold, "Number of instructions seen before we cleanup the value table", DEFAULT\_CONFIG\_GoptCleanupThreshold)

FLAGNR(Number, AsmGoptCleanupThreshold, "Number of instructions seen before we cleanup the value table", DEFAULT\_CONFIG\_AsmGoptCleanupThreshold)

FLAGNR(Boolean, HighPrecisionDate, "Enable sub-millisecond resolution in Javascript Date for benchmark timing", DEFAULT\_CONFIG\_HighPrecisionDate)

FLAGNR(Number, InlineCountMax , "Maximum count in bytecodes to inline in a given function", DEFAULT\_CONFIG\_InlineCountMax)

FLAGNR(Number, InlineThreshold , "Maximum size in bytecodes of an inline candidate", DEFAULT\_CONFIG\_InlineThreshold)

FLAGNR(Number, AggressiveInlineCountMax, "Maximum count in bytecodes to inline in a given function", DEFAULT\_CONFIG\_AggressiveInlineCountMax)

FLAGNR(Number, AggressiveInlineThreshold, "Maximum size in bytecodes of an inline candidate for aggressive inlining", DEFAULT\_CONFIG\_AggressiveInlineThreshold)

FLAGNR(Number, InlineThresholdAdjustCountInLargeFunction , "Adjustment in the maximum size in bytecodes of an inline candidate in a large function", DEFAULT\_CONFIG\_InlineThresholdAdjustCountInLargeFunction)

FLAGNR(Number, InlineThresholdAdjustCountInMediumSizedFunction , "Adjustment in the maximum size in bytecodes of an inline candidate in a medium sized function", DEFAULT\_CONFIG\_InlineThresholdAdjustCountInMediumSizedFunction)

FLAGNR(Number, InlineThresholdAdjustCountInSmallFunction , "Adjustment in the maximum size in bytecodes of an inline candidate in a small function", DEFAULT\_CONFIG\_InlineThresholdAdjustCountInSmallFunction)

FLAGNR(String, Interpret , "List of functions to interpret", nullptr)

FLAGNR(Phases, Instrument , "Instrument the generated code from the given phase", )

FLAGNR(Number, JitQueueThreshold , "Max number of work items/script context in the jit queue", DEFAULT\_CONFIG\_JitQueueThreshold)

#ifdef LEAK\_REPORT

FLAGNR(String, LeakReport , "File name for the leak report", nullptr)

#endif

FLAGNR(Number, LoopInlineThreshold , "Maximum size in bytecodes of an inline candidate with loops or not enough profile data", DEFAULT\_CONFIG\_LoopInlineThreshold)

FLAGNR(Number, LeafInlineThreshold , "Maximum size in bytecodes of an inline candidate with loops or not enough profile data", DEFAULT\_CONFIG\_LeafInlineThreshold)

FLAGNR(Number, ConstantArgumentInlineThreshold, "Maximum size in bytecodes of an inline candidate with constant argument and the argument being used for a branch", DEFAULT\_CONFIG\_ConstantArgumentInlineThreshold)

FLAGNR(Number, RecursiveInlineThreshold, "Maximum size in bytecodes of an inline candidate to inline recursively", DEFAULT\_CONFIG\_RecursiveInlineThreshold)

FLAGNR(Number, RecursiveInlineDepthMax, "Maximum depth of a recursive inline call", DEFAULT\_CONFIG\_RecursiveInlineDepthMax)

FLAGNR(Number, RecursiveInlineDepthMin, "Maximum depth of a recursive inline call", DEFAULT\_CONFIG\_RecursiveInlineDepthMin)

FLAGNR(Number, Loop , "Number of times to execute the script (useful for profiling short benchmarks and finding leaks)", DEFAULT\_CONFIG\_Loop)

FLAGRA(Number, LoopInterpretCount , lic, "Number of times loop has to be interpreted before JIT Loop body", DEFAULT\_CONFIG\_LoopInterpretCount)

FLAGNR(Number, LoopProfileIterations , "Number of iterations of a loop that must be profiled before jitting the loop body", DEFAULT\_CONFIG\_LoopProfileIterations)

FLAGNR(Number, OutsideLoopInlineThreshold , "Maximum size in bytecodes of an inline candidate outside a loop in inliner", DEFAULT\_CONFIG\_OutsideLoopInlineThreshold)

FLAGNR(Number, MaxFuncInlineDepth , "Number of times to allow inlining a function recursively, plus one (min: 1, max: 255)", DEFAULT\_CONFIG\_MaxFuncInlineDepth)

FLAGNR(Number, MaxNumberOfInlineesWithLoop, "Number of times to allow inlinees with a loop in a top function", DEFAULT\_CONFIG\_MaxNumberOfInlineesWithLoop)

#ifdef MEMSPECT\_TRACKING

FLAGNR(Phases, Memspect, "Enables memspect tracking to perform memory investigations.", )

#endif

FLAGNR(Number, PolymorphicInlineThreshold , "Maximum size in bytecodes of an polymorphic inline candidate", DEFAULT\_CONFIG\_PolymorphicInlineThreshold)

FLAGNR(Boolean, PrimeRecycler , "Prime the recycler first", DEFAULT\_CONFIG\_PrimeRecycler)

#if defined(CHECK\_MEMORY\_LEAK) || defined(LEAK\_REPORT)

FLAGNR(Boolean, LeakStackTrace , "Include stack trace on leaked pinned object and heap objects", false)

FLAGNR(Boolean, ForceMemoryLeak , "Fake leak some memory to test leak report and check memory leak", false)

#endif

FLAGNR(Boolean, TDZ , "Emit temporal dead zone code for let/const", DEFAULT\_CONFIG\_TDZ)

FLAGNR(Boolean, ForceOldDateAPI , "Force Chakra to use old dates API regardless of availability of a new one", DEFAULT\_CONFIG\_ForceOldDateAPI)

FLAGNR(Number, JitLoopBodyHotLoopThreshold , "Number of times loop has to be iterated in jitloopbody before it is determined as hot", DEFAULT\_CONFIG\_JitLoopBodyHotLoopThreshold)

FLAGNR(Number, LoopBodySizeThresholdToDisableOpts, "Minimum bytecode size of a loop body, above which we might consider switching off optimizations in jit loop body to avoid rejits", DEFAULT\_CONFIG\_LoopBodySizeThresholdToDisableOpts)

FLAGNR(Number, MaxJitThreadCount , "Number of maximum allowed parallel jit threads (actual number is factor of number of processors and other heuristics)", DEFAULT\_CONFIG\_MaxJitThreadCount)

FLAGNR(Boolean, ForceMaxJitThreadCount, "Force the number of parallel jit threads as specified by MaxJitThreadCount flag (creation guaranteed)", DEFAULT\_CONFIG\_ForceMaxJitThreadCount)

FLAGNR(Number, MinInterpretCount , "Minimum number of times a function must be interpreted", 0)

FLAGNR(Number, MinSimpleJitRunCount , "Minimum number of times a function must be run in simple jit", 0)

FLAGNRA(Number, MaxInterpretCount , Mic, "Maximum number of times a function can be interpreted", 0)

FLAGNRA(Number, MaxSimpleJitRunCount , Msjrc, "Maximum number of times a function will be run in SimpleJitted code", 0)

FLAGNRA(Number, MinMemOpCount , Mmoc, "Minimum count of a loop to activate MemOp", DEFAULT\_CONFIG\_MinMemOpCount)

#if ENABLE\_COPYONACCESS\_ARRAY

FLAGNR(Number, MaxCopyOnAccessArrayLength, "Maximum length of copy-on-access array", DEFAULT\_CONFIG\_MaxCopyOnAccessArrayLength)

FLAGNR(Number, MinCopyOnAccessArrayLength, "Minimum length of copy-on-access array", DEFAULT\_CONFIG\_MinCopyOnAccessArrayLength)

FLAGNR(Number, CopyOnAccessArraySegmentCacheSize, "Size of copy-on-access array segment cache (1-32)", DEFAULT\_CONFIG\_CopyOnAccessArraySegmentCacheSize)

#endif

FLAGNR(Number, MinTemplatizedJitRunCount, "Minimum number of times a function must be Templatized Jitted", DEFAULT\_CONFIG\_MinTemplatizedJitRunCount)

FLAGNR(Number, MinAsmJsInterpreterRunCount, "Minimum number of times a function must be Asm Interpreted", DEFAULT\_CONFIG\_MinAsmJsInterpreterRunCount)

FLAGNR(Number, MinTemplatizedJitLoopRunCount, "Minimum LoopCount run of the Templatized Jit function to run FullJited", DEFAULT\_CONFIG\_MinTemplatizedJitLoopRunCount)

FLAGNRA(Number, MaxTemplatizedJitRunCount, Mtjrc, "Maximum number of times a function must be templatized jit", DEFAULT\_CONFIG\_MaxTemplatizedJitRunCount)

FLAGNRA(Number, MaxAsmJsInterpreterRunCount, Maic, "Maximum number of times a function must be interpreted in asmjs", DEFAULT\_CONFIG\_MaxAsmJsInterpreterRunCount)

FLAGR (Number, AutoProfilingInterpreter0Limit, "Limit after which to transition to the next execution mode", DEFAULT\_CONFIG\_AutoProfilingInterpreter0Limit)

FLAGR (Number, ProfilingInterpreter0Limit, "Limit after which to transition to the next execution mode", DEFAULT\_CONFIG\_ProfilingInterpreter0Limit)

FLAGR (Number, AutoProfilingInterpreter1Limit, "Limit after which to transition to the next execution mode", DEFAULT\_CONFIG\_AutoProfilingInterpreter1Limit)

FLAGR (Number, SimpleJitLimit, "Limit after which to transition to the next execution mode", DEFAULT\_CONFIG\_SimpleJitLimit)

FLAGR (Number, ProfilingInterpreter1Limit, "Limit after which to transition to the next execution mode", DEFAULT\_CONFIG\_ProfilingInterpreter1Limit)

FLAGNRA(String, ExecutionModeLimits, Eml, "Execution mode limits in th form: AutoProfilingInterpreter0.ProfilingInterpreter0.AutoProfilingInterpreter1.SimpleJit.ProfilingInterpreter1 - Example: -ExecutionModeLimits:12.4.0.132.12", L"")

FLAGRA(Boolean, EnforceExecutionModeLimits, Eeml, "Enforces the execution mode limits such that they are never exceeded.", false)

FLAGNRA(Number, SimpleJitAfter , Sja, "Number of calls to a function after which to simple-JIT the function", 0)

FLAGNRA(Number, FullJitAfter , Fja, "Number of calls to a function after which to full-JIT the function. The function will be profiled for every iteration.", 0)

FLAGNR(Boolean, NewSimpleJit , "Uses the new simple JIT", DEFAULT\_CONFIG\_NewSimpleJit)

FLAGNR(Number, MaxLinearIntCaseCount , "Maximum number of cases(in switch statement) for which instructions can be generated linearly",DEFAULT\_CONFIG\_MaxLinearIntCaseCount)

FLAGNR(Number, MaxSingleCharStrJumpTableSize, "Maximum single char string jump table size", DEFAULT\_CONFIG\_MaxSingleCharStrJumpTableSize)

FLAGNR(Number, MaxSingleCharStrJumpTableRatio, "Maximum single char string jump table size as multiples of the actual case arm", DEFAULT\_CONFIG\_MaxSingleCharStrJumpTableRatio)

FLAGNR(Number, MinSwitchJumpTableSize , "Minimum size of the jump table, that is created for consecutive integer case arms in a Switch Statement",DEFAULT\_CONFIG\_MinSwitchJumpTableSize)

FLAGNR(Number, MaxLinearStringCaseCount, "Maximum number of string cases(in switch statement) for which instructions can be generated linearly",DEFAULT\_CONFIG\_MaxLinearStringCaseCount)

FLAGR(Number, MinDeferredFuncTokenCount, "Minimum length in tokens of defer-parsed function", DEFAULT\_CONFIG\_MinDeferredFuncTokenCount)

#if DBG

FLAGNR(Number, SkipFuncCountForBailOnNoProfile, "Initial Number of functions in a func body to be skipped from forcibly inserting BailOnNoProfile.", DEFAULT\_CONFIG\_SkipFuncCountForBailOnNoProfile)

#endif

FLAGNR(Number, MaxJITFunctionBytecodeSize, "The biggest function we'll JIT (bytecode size)", DEFAULT\_CONFIG\_MaxJITFunctionBytecodeSize)

FLAGNR(Number, MaxLoopsPerFunction , "Maximum number of loops in any function in the script", DEFAULT\_CONFIG\_MaxLoopsPerFunction)

FLAGNR(Number, FuncObjectInlineCacheThreshold , "Maximum number of inline caches a function body may have to allow for inline caches to be allocated on the function object", DEFAULT\_CONFIG\_FuncObjectInlineCacheThreshold)

FLAGNR(Boolean, NoDeferParse , "Disable deferred parsing", false)

FLAGNR(Boolean, NoLogo , "No logo, which we don't display anyways", false)

#ifdef \_ARM64\_

FLAGR (Boolean, NoNative , "Disable native codegen", true)

#else

FLAGR (Boolean, NoNative , "Disable native codegen", false)

#endif

FLAGNR(Number, NopFrequency , "Frequency of NOPs inserted by NOP insertion phase. A NOP is guaranteed to be inserted within a range of (1<<n) instrs (default=8)", DEFAULT\_CONFIG\_NopFrequency)

FLAGNR(Boolean, NoStrictMode , "Disable strict mode checks on all functions", false)

FLAGNR(Boolean, NormalizeStats , "When dumping stats, do some normalization (used with -instrument:linearscan)", false)

#ifdef ENABLE\_PROJECTION

FLAGNR(Boolean, NoWinRTFastSig , "Disable fast call for common WinRT function signatures", false)

#endif

FLAGNR(Phases, Off , "Turn off specific phases or feature.(Might not work for all phases)", )

FLAGNR(Phases, OffProfiledByteCode , "Turn off specific byte code for phases or feature.(Might not work for all phases)", )

FLAGNR(Phases, On , "Turn on specific phases or feature.(Might not work for all phases)", )

FLAGNR(String, OutputFile , "Log the output to a specified file. Default: output.log in the working directory.", L"output.log")

FLAGNR(String, OutputFileOpenMode , "File open mode for OutputFile. Default: wt, specify 'at' for append", L"wt")

#ifdef ENABLE\_TRACE

FLAGNR(Boolean, InMemoryTrace , "Enable in-memory trace (investigate crash using trace in dump file). Use !jd.dumptrace to print it.", DEFAULT\_CONFIG\_InMemoryTrace)

FLAGNR(Number, InMemoryTraceBufferSize, "The size of circular buffer for in-memory trace (the units used is: number of trace calls). ", DEFAULT\_CONFIG\_InMemoryTraceBufferSize)

FLAGNR(Boolean, RichTraceFormat, "Whether to use extra data in Output/Trace header.", DEFAULT\_CONFIG\_RichTraceFormat)

#ifdef STACK\_BACK\_TRACE

FLAGNR(Boolean, TraceWithStack, "Whether the trace need to include stack trace (for each trace entry).", DEFAULT\_CONFIG\_TraceWithStack)

#endif STACK\_BACK\_TRACE

#endif ENABLE\_TRACE

FLAGNR(Boolean, PrintRunTimeDataCollectionTrace, "Print traces needed for runtime data collection", false)

#ifdef ENABLE\_PREJIT

FLAGR (Boolean, Prejit , "Prejit everything, including things that are not called, ignoring limits (default: false)", DEFAULT\_CONFIG\_Prejit)

#endif

FLAGNR(Boolean, PrintSrcInDump , "Print the lineno and the source code in the intermediate dumps", true)

#if PROFILE\_DICTIONARY

FLAGNR(Number, ProfileDictionary , "Profile dictionary usage. Only dictionaries with max depth of <number> or above are displayed (0=no filter).", -1)

#endif

#ifdef PROFILE\_EXEC

FLAGNR(Phases, Profile , "Profile the given phase", )

FLAGNR(Number, ProfileThreshold , "A phase is displayed in the profiler report only if its contribution is more than this threshold", 0)

#endif

#ifdef PROFILE\_OBJECT\_LITERALS

FLAGNR(Boolean, ProfileObjectLiteral , "Profile Object literal usage", false)

#endif

#ifdef PROFILE\_MEM

FLAGNR(String, ProfileMemory , "Profile memory usage", )

#endif

#ifdef PROFILE\_STRINGS

FLAGNR(Boolean, ProfileStrings , "Profile string statistics", false)

#endif

#ifdef PROFILE\_TYPES

FLAGNR(Boolean, ProfileTypes , "Profile type statistics", false)

#endif

#ifdef PROFILE\_EVALMAP

FLAGNR(Boolean, ProfileEvalMap , "Profile eval map statistics", false)

#endif

#ifdef PROFILE\_BAILOUT\_RECORD\_MEMORY

FLAGNR(Boolean, ProfileBailOutRecordMemory, "Profile bailout record memory statistics", false)

#endif

FLAGNR(Number, RejitMaxBailOutCount, "Maximum number of bailouts for a bailout record after which rejit is forced", DEFAULT\_CONFIG\_RejitMaxBailOutCount)

FLAGNR(Number, RejitRatioLimit, "Rejit ratio (Percentage of bailouts per function after which rejit is queued)", DEFAULT\_CONFIG\_RejitRatioLimit)

FLAGNR(Number, MinBailOutsBeforeRejit, "Minimum number of bailouts for a single bailout record after which a rejit is considered", DEFAULT\_CONFIG\_MinBailOutsBeforeRejit)

FLAGNR(Boolean, LibraryStackFrame , "Display library stack frame", DEFAULT\_CONFIG\_LibraryStackFrame)

FLAGNR(Boolean, LibraryStackFrameDebugger , "Assume debugger support for library stack frame", DEFAULT\_CONFIG\_LibraryStackFrameDebugger)

#ifdef RECYCLER\_STRESS

FLAGNR(Boolean, RecyclerStress , "Stress the recycler by collect on every allocation call", false)

#ifdef CONCURRENT\_GC\_ENABLED

FLAGNR(Boolean, RecyclerBackgroundStress , "Stress the recycler by collect in the background thread on every allocation call", false)

FLAGNR(Boolean, RecyclerConcurrentStress , "Stress the concurrent recycler by concurrent collect on every allocation call", false)

FLAGNR(Boolean, RecyclerConcurrentRepeatStress , "Stress the concurrent recycler by concurrent collect on every allocation call and repeat mark and rescan in the background thread", false)

#endif

#ifdef PARTIAL\_GC\_ENABLED

FLAGNR(Boolean, RecyclerPartialStress , "Stress the partial recycler by partial collect on every allocation call", false)

#endif

FLAGNR(Boolean, RecyclerTrackStress, "Stress tracked object handling by simulating tracked objects for regular allocations", false)

FLAGNR(Boolean, RecyclerInduceFalsePositives, "Stress recycler by forcing false positive object marks", false)

#endif // RECYCLER\_STRESS

FLAGNR(Boolean, RecyclerForceMarkInterior, "Force all the mark as interior", DEFAULT\_CONFIG\_RecyclerForceMarkInterior)

#ifdef CONCURRENT\_GC\_ENABLED

FLAGNR(Number, RecyclerPriorityBoostTimeout, "Adjust priority boost timeout", 5000)

FLAGNR(Number, RecyclerThreadCollectTimeout, "Adjust thread collect timeout", 1000)

#endif

#ifdef RECYCLER\_PAGE\_HEAP

FLAGNR(Number, PageHeap, "Use full page for heap allocations", DEFAULT\_CONFIG\_PageHeap)

FLAGNR(Boolean, PageHeapAllocStack, "Capture alloc stack under page heap mode", DEFAULT\_CONFIG\_PageHeapAllocStack)

FLAGNR(Boolean, PageHeapFreeStack, "Capture free stack under page heap mode", DEFAULT\_CONFIG\_PageHeapFreeStack)

FLAGNR(NumberRange, PageHeapBucketNumber, "Bucket numbers to be used for page heap allocations", )

FLAGNR(Number, PageHeapBlockType, "Type of blocks to use page heap for", DEFAULT\_CONFIG\_PageHeapBlockType)

#endif

#ifdef RECYCLER\_NO\_PAGE\_REUSE

FLAGNR(Boolean, RecyclerNoPageReuse, "Do not reuse page in recycler", false)

#endif

#ifdef RECYCLER\_MEMORY\_VERIFY

FLAGNR(Phases, RecyclerVerify , "Verify recycler memory", )

FLAGNR(Number, RecyclerVerifyPadSize , "Padding size to verify recycler memory", 12)

#endif

FLAGNR(Boolean, RecyclerTest , "Run recycler tests instead of executing script", false)

FLAGNR(Boolean, RecyclerProtectPagesOnRescan, "Temporarily switch all pages to read only during rescan", false)

#ifdef RECYCLER\_VERIFY\_MARK

FLAGNR(Boolean, RecyclerVerifyMark , "verify concurrent gc", false)

#endif

FLAGR (Number, LowMemoryCap , "Memory cap indicating a low-memory process", DEFAULT\_CONFIG\_LowMemoryCap)

#ifdef RUNTIME\_DATA\_COLLECTION

FLAGNR(String, RuntimeDataOutputFile, "Filename to write the dynamic profile info", nullptr)

#endif

FLAGNR(Boolean, ReportErrors , "Enable reporting of syntax errors", false)

FLAGR (Number, SpeculationCap , "How much bytecode we'll speculatively JIT", DEFAULT\_CONFIG\_SpeculationCap)

#if DBG\_DUMP || defined(BGJIT\_STATS) || defined(RECYCLER\_STATS)

FLAGNR(Phases, Stats , "Stats the given phase", )

#endif

#if EXCEPTION\_RECOVERY

FLAGNR(Boolean, SwallowExceptions , "Force a try/catch around every statement", false)

#endif

FLAGNR(Boolean, PrintSystemException , "Always print a message when there's OOM or OOS", false)

FLAGNR(Number, SwitchOptHolesThreshold, "Maximum percentage of holes (missing case values in a switch statement) with which a jump table can be created",DEFAULT\_CONFIG\_SwitchOptHolesThreshold)

FLAGR (Number, TempMin , "Temp number switch which code can temporarily use for debugging", DEFAULT\_CONFIG\_TempMin)

FLAGR (Number, TempMax , "Temp number switch which code can temporarily use for debugging", DEFAULT\_CONFIG\_TempMax)

FLAGNR(Phases, Trace , "Trace the given phase", )

#if defined(\_M\_IX86) || defined(\_M\_X64)

FLAGR(Number, LoopAlignNopLimit , "Max number of nops for loop alignment", DEFAULT\_CONFIG\_LoopAlignNopLimit)

#endif

#ifdef PROFILE\_MEM

FLAGNR(Phases, TraceMemory , "Trace memory usage", )

#endif

#if DBG\_DUMP || defined(RECYCLER\_TRACE)

//TraceMetaDataParsing flag with optional levels:

// Level 1 = interfaces only

// Level 2 = interfaces and methods

// Level 3 = interfaces, methods and parameters

// Level 4 = interfaces and properties

// Level 5 (default) = ALL

FLAGNR(Number, TraceMetaDataParsing , "Trace metadata parsing for generating JS projections. [Levels 1-5, with 5 corresponding to most detailed]", 5)

FLAGNR(Boolean, TraceWin8Allocations , "Trace the win8 memory allocations", false)

FLAGNR(Boolean, TraceWin8DeallocationsImmediate , "Trace the win8 memory deallocations immediately", false)

FLAGNR(Boolean, PrintWin8StatsDetailed , "Print the detailed memory trace report", false)

FLAGNR(Boolean, TraceProtectPages , "Trace calls to protecting pages of custom heap allocated pages", false)

//TraceProjection flag with optional levels:

// Level 1 = error

// Level 2 = warning

// Level 3 = informational

FLAGNR(Number, TraceProjection , "Trace projection related activities, [Levels 1-3, with 3 corresponding to most detailed]", 3)

#endif

FLAGNR(Boolean, TraceAsyncDebugCalls , "Trace calls to async debugging API (default: false)", DEFAULT\_CONFIG\_TraceAsyncDebugCalls)

#ifdef TRACK\_DISPATCH

FLAGNR(Boolean, TrackDispatch , "Save stack traces of where JavascriptDispatch/HostVariant are created", false)

#endif

FLAGNR(Boolean, Verbose , "Dump details", DEFAULT\_CONFIG\_Verbose)

FLAGNR(Boolean, UseFullName , "Enable fully qualified name", DEFAULT\_CONFIG\_UseFullName)

FLAGNR(Boolean, UseFunctionIdForTrace , "Use function id instead of function number for trace output", false)

FLAGNR(Boolean, Utf8 , "Use UTF8 for file output", false)

FLAGR (Number, Version , "Version in which to run the jscript engine. [one of 1,2,3,4,5,6]. Default is latest for jc/jshost, 1 for IE", 6 )

#ifdef ENABLE\_PROJECTION

FLAGR (Number, HostType , "Host type in which to run the jscript engine. [one of 1,2]. Default is 1 = Browser.", 1)

#endif

FLAGR (Boolean, WERExceptionSupport , "WER feature for extended exception support. Enabled when WinRT is enabled", false )

#ifdef ENABLE\_PROJECTION

FLAGR (Boolean, WinRTConstructorAllowed, "Whether WinRT constructors is allowed in WebView host type. Constructor is always allowed in other host type ", false)

#endif

FLAGNR(Boolean, ExtendedErrorStackForTestHost, "Enable passing extended error stack string to test host.", DEFAULT\_CONFIG\_ExtendedErrorStackForTestHost)

FLAGNR(Boolean, errorStackTrace , "error.StackTrace feature. Remove when feature complete", DEFAULT\_CONFIG\_errorStackTrace)

FLAGNR(Boolean, DoHeapEnumOnEngineShutdown, "Perform a heap enumeration whenever shut a script engine down", false)

#ifdef HEAP\_ENUMERATION\_VALIDATION

FLAGNR(Boolean, ValidateHeapEnum , "Validate that heap enumeration is reporting all Js::RecyclableObjects in the heap", false)

#endif

#if ENABLE\_REGEX\_CONFIG\_OPTIONS

//

// Regex flags

//

FLAGR (Boolean, RegexTracing , "Trace all Regex invocations to the output.", DEFAULT\_CONFIG\_RegexTracing)

FLAGR (Boolean, RegexProfile , "Collect usage statistics on all Regex invocations.", DEFAULT\_CONFIG\_RegexProfile)

FLAGR (Boolean, RegexDebug , "Trace compilation of UnifiedRegex expressions.", DEFAULT\_CONFIG\_RegexDebug)

FLAGR (Boolean, RegexOptimize , "Optimize regular expressions in the unified Regex system (default: true)", DEFAULT\_CONFIG\_RegexOptimize)

FLAGR (Number, DynamicRegexMruListSize, "Size of the MRU list for dynamic regexes", DEFAULT\_CONFIG\_DynamicRegexMruListSize)

#endif

FLAGR (Boolean, OptimizeForManyInstances, "Optimize script engine for many instances (low memory footprint per engine, assume low spare CPU cycles) (default: false)", DEFAULT\_CONFIG\_OptimizeForManyInstances)

FLAGNR(Phases, TestTrace , "Test trace for the given phase", )

FLAGNR(Boolean, EnableEvalMapCleanup, "Enable cleaning up the eval map", true)

#ifdef PROFILE\_MEM

FLAGNR(Boolean, TraceObjectAllocation, "Enable cleaning up the eval map", false)

#endif

#ifdef ENABLE\_PROJECTION

FLAGNR(Boolean, EnableThirdPartyGCPressure, "Enable use of GCPressure attribute value on 3rd party WinRT objects (not in Windows namespace) (default: false)", false)

FLAGNR(Number, TargetWinRTVersion, "Specifies WinRT version number to target. [one of 0,1,2,3,4]. Default is 1 = NTDDI\_WIN8", DEFAULT\_CONFIG\_TargetWinRTVersion)

FLAGNR(Boolean, EnableVersioningAllAssemblies, "Enable versioning behavior for all assemblies, regardless of host flag (default: false)", false)

FLAGR(Boolean, FailFastIfDisconnectedDelegate, "When set fail fast if disconnected delegate is invoked", DEFAULT\_CONFIG\_FailFastIfDisconnectedDelegate)

#endif

FLAGNR(Number, Sse, "Virtually disables SSE-based optimizations above the specified SSE level in the Chakra JIT (does not affect CRT SSE usage)", DEFAULT\_CONFIG\_Sse)

FLAGNR(Number, DeletedPropertyReuseThreshold, "Start reusing deleted property indexes after this many properties are deleted. Zero to disable reuse.", DEFAULT\_CONFIG\_DeletedPropertyReuseThreshold)

FLAGNR(Boolean, ForceStringKeyedSimpleDictionaryTypeHandler, "Force switch to string keyed version of SimpleDictionaryTypeHandler on first new property added to a SimpleDictionaryTypeHandler", DEFAULT\_CONFIG\_ForceStringKeyedSimpleDictionaryTypeHandler)

FLAGNR(Number, BigDictionaryTypeHandlerThreshold, "Min Slot Capacity required to convert DictionaryTypeHandler to BigDictionaryTypeHandler.(Advisable to give more than 15 - to avoid false positive cases)", DEFAULT\_CONFIG\_BigDictionaryTypeHandlerThreshold)

FLAGNR(Boolean, TypeSnapshotEnumeration, "Create a true snapshot of the type of an object before enumeration and enumerate only those properties.", DEFAULT\_CONFIG\_TypeSnapshotEnumeration)

FLAGR (Boolean, EnumerationCompat, "When set in IE10 mode, restores enumeration behavior to RC behavior", DEFAULT\_CONFIG\_EnumerationCompat)

FLAGNR(Boolean, IsolatePrototypes, "Should prototypes get unique types not shared with other objects (default: true)?", DEFAULT\_CONFIG\_IsolatePrototypes)

FLAGNR(Boolean, ChangeTypeOnProto, "When becoming a prototype should the object switch to a new type (default: true)?", DEFAULT\_CONFIG\_ChangeTypeOnProto)

FLAGNR(Boolean, ShareInlineCaches, "Determines whether inline caches are shared between all loads (or all stores) of the same property ID", DEFAULT\_CONFIG\_ShareInlineCaches)

FLAGNR(Boolean, DisableDebugObject, "Disable test only Debug object properties", DEFAULT\_CONFIG\_DisableDebugObject)

FLAGNR(Boolean, DumpHeap, "enable Debug.dumpHeap even when DisableDebugObject is set", DEFAULT\_CONFIG\_DumpHeap)

FLAGNR(String, autoProxy, "enable creating proxy for each object creation", L"\_\_msTestHandler")

FLAGNR(Number, PerfHintLevel, "Specifies the perf-hint level (1,2) 1 == critical, 2 == only noisy", DEFAULT\_CONFIG\_PerfHintLevel)

#ifdef INTERNAL\_MEM\_PROTECT\_HEAP\_ALLOC

FLAGNR(Boolean, MemProtectHeap, "Use the mem protect heap as the default heap", DEFAULT\_CONFIG\_MemProtectHeap)

#endif

#ifdef RECYCLER\_STRESS

FLAGNR(Boolean, MemProtectHeapStress, "Stress the recycler by collect on every allocation call", false)

#ifdef CONCURRENT\_GC\_ENABLED

FLAGNR(Boolean, MemProtectHeapBackgroundStress, "Stress the recycler by collect in the background thread on every allocation call", false)

FLAGNR(Boolean, MemProtectHeapConcurrentStress, "Stress the concurrent recycler by concurrent collect on every allocation call", false)

FLAGNR(Boolean, MemProtectHeapConcurrentRepeatStress, "Stress the concurrent recycler by concurrent collect on every allocation call and repeat mark and rescan in the background thread", false)

#endif

#ifdef PARTIAL\_GC\_ENABLED

FLAGNR(Boolean, MemProtectHeapPartialStress, "Stress the partial recycler by partial collect on every allocation call", false)

#endif

#endif

#ifdef SUPPORT\_FIXED\_FIELDS\_ON\_PATH\_TYPES

FLAGNR(Boolean, FixPropsOnPathTypes, "Mark properties as fixed on path types (default: false).", DEFAULT\_CONFIG\_FixPropsOnPathTypes)

#endif

FLAGNR(NumberSet, BailoutTraceFilter, "Filter the bailout trace messages to specific bailout kinds.", )

FLAGNR(NumberSet, RejitTraceFilter, "Filter the rejit trace messages to specific bailout kinds.", )

// recycler heuristic flags

FLAGNR(Number, MaxBackgroundFinishMarkCount, "Maximum number of background finish mark", 1)

FLAGNR(Number, BackgroundFinishMarkWaitTime, "Millisecond to wait for background finish mark", 15)

FLAGNR(Number, MinBackgroundRepeatMarkRescanBytes, "Minimum number of bytes rescan to trigger background finish mark", -1)

// recycler memory restrict test flags

FLAGNR(Number, MaxMarkStackPageCount , "Restrict recycler mark stack size (in pages)", -1)

FLAGNR(Number, MaxTrackedObjectListCount, "Restrict recycler tracked object count during GC", -1)

#if DBG

FLAGNR(Boolean, InitializeInterpreterSlotsWithInvalidStackVar, "Enable the initialization of the interpreter local slots with invalid stack vars", false)

#endif

FLAGNR(Boolean, ClearInlineCachesOnCollect, "Clear all inline caches on every garbage collection", false)

FLAGNR(Number, InlineCacheInvalidationListCompactionThreshold, "Compact inline cache invalidation lists if their utilization falls below this threshold", DEFAULT\_CONFIG\_InlineCacheInvalidationListCompactionThreshold)

#ifdef IR\_VIEWER

FLAGNR(Boolean, IRViewer, "Enable IRViewer functionality (improved UI for various stages of IR generation)", false)

#endif /\* IR\_VIEWER \*/

FLAGNR(Boolean, InvalidateSolutionContextsForGetStructure, "To reduce memory consumption, in the end of GetStructure call, invalidate script contexts used only for GetStructure -- this would invalidate ones associated with solution files (not top-most references such as helpers.js)", DEFAULT\_CONFIG\_InvalidateSolutionContextsForGetStructure)

FLAGNR(Boolean, GCPauseTel, "Enable GC Pause telemetry in the product code.", false)

FLAGNR(Boolean, ES5LangTel, "Print ES5 language telemetry output.", false)

FLAGNR(Boolean, ES6LangTel, "Print ES6 language telemetry output.", false)

FLAGNR(Boolean, ESBLangTel, "Print ES built-ins telemetry output.", false)

FLAGNR(Boolean, DateParseTel, "Print Date.parse telemetry output.", false)

FLAGNR(Number, GCMemoryThreshold, "Threshold for allocation-based GC initiation (in MB)", 0)

#ifdef \_CONTROL\_FLOW\_GUARD

FLAGNR(Boolean, PreReservedHeapAlloc, "Enable Pre-reserved Heap Page Allocator", true)

FLAGNR(Boolean, CFGInJit, "Enable CFG check in JIT", true)

FLAGNR(Boolean, CFG, "Force enable CFG on jshost. version in the jshost's manifest file disables CFG", true)

#endif

#if DBG

FLAGNR(Number, SimulatePolyCacheWithOneTypeForInlineCacheIndex, "Use with SimulatePolyCacheWithOneTypeForFunction to simulate creating a polymorphic inline cache containing only one type due to a collision, for testing ObjTypeSpec", -1)

#endif

#undef FLAG\_REGOVR\_EXP

#undef FLAG\_REGOVR\_ASMJS

#undef FLAG

#undef FLAGP

#undef FLAGRA

#undef FLAGNR

#undef FLAGNRA

#undef FLAGPNR

#endif