OVP VMI/OP Function Overview

Imperas Software Limited

Imperas Buildings, North Weston,
Thame, Oxfordshire, OX9 2HA, UK
docs@imperas.com

Author:	Imperas Software Limited
Version:	1.0.1
Filename:	OVP_VMI_OP_Function_Overview.doc
Project:	OVP VMI/OP Function Overview
Last Saved:	Tuesday, 18 May 2021
Keywords:	

Copyright Notice

Copyright © 2021 Imperas Software Limited All rights reserved. This software and documentation contain information that is the property of Imperas Software Limited. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Imperas Software Limited, or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Imperas permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

IMPERAS SOFTWARE LIMITED, AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1 Introduction **5**

2 VMI Run Time Functions **6**

2.1 SIMULATION ENVIRONMENT ACCESS 6
2.2 PROCESSOR SIMULATION CONTROL 6
2.3 PROGRAM COUNTER AND CODE DICTIONARY 6
2.4 DICTIONARY AND BLOCK MODES 7
2.5 TIME AND CYCLE COUNTS 7
2.5.1 Instruction/Cycle Counting and Interrupt 7
2.5.2 Simulated Time 7
2.5.3 Delay Estimation 8
2.6 PROCESSOR CONNECTIONS AND REGISTERS 8
2.6.1 Register Access 8
2.6.2 Bus Port Access 8
2.6.3 Net Port Access 8
2.6.4 FIFO Port Access 8
2.6.5 Exception Access 9
2.6.6 Connection Objects 9
2.7 MEMORY OPERATIONS 9
2.7.1 Generic Load/Store 9
2.7.2 Memory Callbacks 9
2.7.3 Memory Manipulation 10
2.7.4 ASID Memory Management 10
2.8 FLOATING POINT 11
2.8.1 Floating-Point Operation Control 11
2.8.2 Floating-Point Operations 11
2.9 SMP PROCESSOR HIERARCHY 11
2.10 OBJECT FILE ACCESS 11
2.10.1 Translation between Object File Addresses and Names 11
2.10.2 Add and Access Object Files 12
2.10.3 Query Object File Symbols 12
2.10.4 Query Object File File/Line 12
2.11 RANGE TABLE HASH 12
2.12 SHARED DATA 13
2.13 NOTIFIERS 13
2.13.1 Register Update Notifiers 13
2.13.2 Branch Reason Notifiers 13
2.14 SAVE/RESTORE SUPPORT 13
2.15 DEBUG VIEW SUPPORT 13
2.16 INSTRUCTION ATTRIBUTES 14
2.17 SHARED OBJECT / DYNAMIC LINKED LIBRARY LOADING 14
2.18 COMMAND INTERPRETER 14
2.19 DEBUGGER INTEGRATION 14
2.20 TRACE INTEGRATION 14
2.21 DOCUMENTATION 15
2.22 MESSAGES 15
2.23 HTTP INTERFACE 15
2.24 LICENSING 15
2.25 ENCAPSULATED MODEL 15

3 OP Functions **16**

3.1 SIMULATION 16
3.1.1 Session Control 16

3.1.2 Execution Control ... 16
3.1.3 Environment Access ... 16
3.1.4 Module Simulation ... 16
3.2 PROGRAM COUNTER AND CODE DICTIONARY ... 17
3.3 TIME AND CYCLE COUNTS ... 17
3.3.1 Instruction/Cycle Counting and Interrupt ... 17
3.3.2 Simulated Time ... 17
3.3.3 Delay Estimation ... 17
3.4 PLATFORM COMPONENT CREATION, CONNECTION AND QUERY ... 18
3.4.1 Bus Bridges ... 18
3.4.2 Buses ... 18
3.4.3 Bus Slaves ... 18
3.4.4 Bus Ports ... 18
3.4.5 Bus Port Connections ... 18
3.4.6 Extensions ... 19
3.4.7 FIFOs ... 19
3.4.8 FIFO Ports ... 19
3.4.9 FIFO Port Connections ... 19
3.4.10 MMCs ... 19
3.4.11 Memory-Mapped Registers ... 19
3.4.12 Memories ... 20
3.4.13 Modules ... 20
3.4.14 Nets ... 20
3.4.15 Net Ports ... 20
3.4.16 Net Port Connections ... 21
3.4.17 Objects ... 21
3.4.18 Packet Nets ... 21
3.4.19 Packet Net Port Connections ... 21
3.4.20 Peripherals ... 22
3.4.21 Processors ... 22
3.4.22 VLNV ... 23
3.5 MEMORY OPERATIONS ... 23
3.5.1 Generic Load/Store ... 23
3.5.2 Memory Callbacks ... 23
3.5.3 Memory Manipulation ... 23
3.6 SMP PROCESSOR HIERARCHY ... 24
3.7 APPLICATION FILE ACCESS ... 24
3.8 SHARED DATA ... 24
3.9 PROCESSOR REGISTERS, EXCEPTIONS AND MODES ... 25
3.9.1 Processor Registers ... 25
3.9.2 Mode Access ... 25
3.9.3 Exception Access ... 25
3.10 PARAMETERS ... 26
3.10.1 Formal Parameters ... 26
3.10.2 Actual Parameters ... 26
3.11 SAVE/RESTORE SUPPORT ... 27
3.12 INSTRUCTION ATTRIBUTES ... 27
3.13 COMMAND INTERPRETER ... 27
3.14 DEBUGGER INTEGRATION ... 28
3.15 BREAKPOINTS ... 28
3.16 WATCHPOINTS ... 28
3.17 TRIGGERS ... 29
3.18 TRACE INTEGRATION ... 29
3.19 DOCUMENTATION ... 29
3.20 MESSAGES ... 29
3.21 HTTP INTERFACE ... 30

1 Introduction

This document provides a high-level overview of the functional groups implemented in the VMI run time and OP interfaces. The intention is not to describe any functions in detail, but to provide a starting point about where to look for functions required to implement a particular task.

The VMI and OP APIs are intended to perform different tasks. The VMI interface is primarily intended for *modeling of processors*. It therefore contains functions for describing instructions to the JIT translation engine, efficient implementation of memory translation schemes, description of processor registers and so on. The OP API is primarily intended for *description of platform interconnect* and *simulation control*. It therefore contains functions for instantiation of components, executing processors and so on. The great majority of tasks that are done when modeling a processor are inappropriate for use in a platform/simulation context and vice-versa, so they are placed in different APIs to guard against misuse.

There is some overlap in the two interfaces when they are used for *tool development*. Tools are sometimes implemented as intercept libraries on processor objects and sometimes in a simulation harness written using the OP API. Therefore, some functions in these two APIs perform similar purposes: this document indicates which functions fall into such groups.

For detailed information on the VMI Run Time function API, consult the *VMI Run Time Function Reference Manual*. For detailed information on the OP API, consult the Doxygen documentation tree in the Imperas installation.

2 VMI Run Time Functions

2.1 Simulation Environment Access

These functions are used to access simulation environment features:

```
vmirtSuppressStdout
vmirtPlatformName
```

2.2 Processor Simulation Control

These functions control inspection and execution of a processor:

vmirtGetCurrentProcessor	opProcessorCurrent
vmirtCPUId	
vmirtGetProcessorForCPUId	
vmirtProcessorFlags	
vmirtProcessorName	
vmirtSetProcessorName	
vmirtProcessorVariant	opProcessorVariant
vmirtSetProcessorVariant	
vmirtProcessorType	
vmirtProcessorStringAttribute	
vmirtProcessorBoolAttribute	
vmirtProcessorUns32Attribute	
vmirtProcessorUns64Attribute	
vmirtProcessorFlt64Attribute	
vmirtGetCurrentMode	opProcessorModeCurrent
vmirtGetNextMode	opProcessorModeNext
vmirtYield	
vmirtHalt	opProcessorHalt
vmirtInterrupt	opInterrupt
vmirtYieldControl	opProcessorYield
vmirtExit	opProcessorExit
vmirtFinish	opProcessorFinish
vmirtStop	opInterruptRSP
vmirtAtomic	
vmirtBlock	
vmirtAbortRepeat	
vmirtIsHalted	
vmirtRestartNext	
vmirtRestartNow	
vmirtDoSynchronousInterrupt	

For related OP functions, see section 3.4.21.

2.3 Program Counter and Code Dictionary

These functions handle access to simulated program counter and invalidation of code dictionary:

vmirtGetPC	opProcessorPC
vmirtGetPCDS	opProcessorPCDS
vmirtSetPC	opProcessorPCSet
vmirtSetPCDS	
vmirtSetPCException	

```
vmirtSetPCFlushTarget
vmirtSetPCFlushDict
vmirtFlushTarget
vmirtFlushTargetMode
vmirtFlushTargetModeTagged
vmirtFlushDict
vmirtFlushAllDicts
vmirtAddPCCallback
vmirtRemovePCCallback
vmirtUpdatePCCallbackCondition
vmirtGetPCCallbackCondition
```

For related OP functions, see section 3.2.

2.4 Dictionary and Block Modes

These functions handle mode-specific JIT code:

```
vmirtGetMode
vmirtSetMode
vmirtGetBlockMask
vmirtSetBlockMask
vmirtSetBlockMask32
vmirtSetBlockMask64
```

2.5 Time and Cycle Counts

2.5.1 Instruction/Cycle Counting and Interrupt

These functions handle instruction and cycle counts:

`vmirtGetProcessorIPS`	
`vmirtGetICount`	`opProcessorCycleCount`
`vmirtGetExecutedICount`	`opProcessorICount`
`vmirtSetICountInterrupt`	
`vmirtClearICountInterrupt`	
`vmirtCreateModelTimer`	
`vmirtCreateMonotonicModelTimer`	
`vmirtDeleteModelTimer`	
`vmirtSetModelTimer`	
`vmirtClearModelTimer`	
`vmirtIsModelTimerEnabled`	
`vmirtGetModelTimerCurrentCount`	
`vmirtGetModelTimerExpiryCount`	

For related OP functions, see section 3.3.1.

2.5.2 Simulated Time

These functions handle interaction with simulated time:

`vmirtGetQuantumStartTime`	
`vmirtGetQuantumEndTime`	
`vmirtGetLocalTime`	`opProcessorTime`
`vmirtGetMonotonicTime`	
`vmirtCreateQuantumTimer`	
`vmirtDeleteQuantumTimer`	

For related OP functions, see section 3.3.2.

2.5.3 Delay Estimation

These functions handle delay estimation:

`vmirtSetDerateFactor`	`opProcessorDerate`
`vmirtGetDerateFactor`	
`vmirtAddSkipCount`	`opProcessorSkipCyclesAdd`
`vmirtGetSkipCount`	`opProcessorSkipCycles`

For related OP functions, see section 3.3.3.

2.6 *Processor Connections and Registers*

2.6.1 Register Access

These functions implement access to processor registers:

`vmirtGetRegGroupByName`	`opProcessorRegGroupByName`
`vmirtGetNextRegGroup`	`opProcessorRegGroupNext`
`vmirtGetRegByName`	`opProcessorRegByName`
`vmirtGetNextReg`	`opProcessorRegNext`
`vmirtGetNextRegInGroup`	`opRegGroupRegNext`
`vmirtRegRead`	`opProcessorRegRead`
`vmirtRegWrite`	`opProcessorRegWrite`

For related OP functions, see section 3.9.1.

2.6.2 Bus Port Access

These functions implement access to processor bus ports:

```
vmirtGetBusPortByName
vmirtGetNextBusPort
```

For related OP functions, see section 3.4.4.

2.6.3 Net Port Access

These functions implement access to processor net ports:

`vmirtGetNetPortByName`	
`vmirtGetNextNetPort`	
`vmirtGetNetPortHandle`	
`vmirtWriteNetPort`	`opNetWrite`
`vmirtReadNetPort`	`opNetValue`
`vmirtInstallNetCallback`	

For related OP functions, see section 3.4.15.

2.6.4 FIFO Port Access

These functions implement access to processor FIFO ports:

```
vmirtGetFifoPortByName
```

```
    vmirtGetNextFifoPort
```

For related OP functions, see section 3.4.8.

2.6.5 Exception Access

These functions implement access to processor exceptions:

```
    vmirtGetCurrentException
    vmirtGetNextException
```

For related OP functions, see section 3.9.3.

2.6.6 Connection Objects

These functions are used to query and update connection objects:

```
    vmirtConnGetInput
    vmirtConnGetOutput
    vmirtConnGetInputInfo
    vmirtConnGetOutputInfo
    vmirtConnGet
    vmirtConnPut
    vmirtConnNotifyGet
    vmirtConnNotifyPut
```

2.7 Memory Operations

2.7.1 Generic Load/Store

These functions implement load and store operations:

```
    vmirtRead1ByteDomain
    vmirtRead2ByteDomain
    vmirtRead4ByteDomain
    vmirtRead8ByteDomain
    vmirtWrite1ByteDomain
    vmirtWrite2ByteDomain
    vmirtWrite4ByteDomain
    vmirtWrite8ByteDomain
    vmirtReadNByteDomain                    opMemoryRead
    vmirtReadNByteDomainVA
    vmirtWriteNByteDomain                   opMemoryWrite
    vmirtWriteNByteDomainVA
    vmirtGetReadNByteSrc
    vmirtGetWriteNByteDst
    vmirtGetString
```

For related OP functions, see section 3.5.1.

2.7.2 Memory Callbacks

These functions handle installation and removal of callback functions on memory accesses:

```
    vmirtAddReadCallback                    opMemoryReadMonitorAdd
    vmirtRemoveReadCallback                 opMemoryReadMonitorDelete
```

vmirtAddWriteCallback	opMemoryWriteMonitorAdd
vmirtRemoveWriteCallback	opMemoryWriteMonitorDelete
vmirtAddFetchCallback	opMemoryFetchMonitorAdd
vmirtRemoveFetchCallback	opMemoryFetchMonitorDelete

For related OP functions, see section 3.5.2.

2.7.3 Memory Manipulation

These functions are used to query and manipulate `memDomain` objects:

vmirtGetProcessorCodeEndian	
vmirtGetProcessorDataEndian	
vmirtGetProcessorCodeDomain	
vmirtGetProcessorDataDomain	
vmirtSetProcessorCodeDomain	
vmirtSetProcessorDataDomain	
vmirtGetProcessorExternalCodeDomain	
vmirtGetProcessorExternalDataDomain	
vmirtGetProcessorInternalCodeDomain	
vmirtGetProcessorInternalDataDomain	
vmirtSetProcessorInternalCodeDomain	
vmirtSetProcessorInternalDataDomain	
vmirtSetProcessorCodeDomains	
vmirtSetProcessorDataDomains	
vmirtIsExecutable	
vmirtSetCreateDomainContext	
vmirtNewDomain	
vmirtGetDomainAddressBits	
vmirtGetDomainPrivileges	
vmirtGetDomainMapped	
vmirtGetNextMappedRange	
vmirtAliasMemory	opDynamicBridge
vmirtAliasMemoryPriv	
vmirtUnaliasMemory	opDynamicUnbridge
vmirtIsAlias	
vmirtMapVAToPA	
vmirtMapToDomain	
vmirtProtectMemory	opBusPrivSet
vmirtMapNativeMemory	opMemoryNativeDynamic
vmirtMapMemory	
vmirtMapCallbacks	
vmirtSetLoadStoreMask	
vmirtDebugDomain	opModuleDomainDebug

For related OP functions, see section 3.5.3.

2.7.4 ASID Memory Management

These functions are used to implement ASID-based memory mappings:

```
vmirtSetProcessorASID
vmirtGetProcessorASID
vmirtAliasMemoryVM
vmirtUnaliasMemoryVM
vmirtGetDomainMappedASID
vmirtGetMRUStateTable
vmirtGetNthStateIndex
```

2.8 Floating Point

2.8.1 Floating-Point Operation Control

These functions are used to control the behavior of floating-point operations:

```
vmirtSetSIMDMaxUnroll
vmirtConfigureFPU
vmirtGetFPControlWord
vmirtSetFPControlWord
vmirtRestoreFPState
```

2.8.2 Floating-Point Operations

These functions are used to implement floating-point operations:

```
vmirtGetFConvertRRDesc
vmirtGetFUnopRRDesc
vmirtGetFBinopRRRDesc
vmirtGetFTernopRRRRDesc
vmirtGetFCompareRRDesc
vmirtGetFCompareRRCDesc
vmirtFConvertSimdRR
vmirtFUnopSimdRR
vmirtFBinopSimdRRR
vmirtFTernopSimdRRRR
vmirtFCompareSimdRR
```

2.9 SMP Processor Hierarchy

These functions are used to traverse SMP processor hierarchy:

`vmirtGetSMPParent`	`opProcessorParent`
`vmirtSetSMPParent`	
`vmirtGetSMPChild`	`opProcessorChild`
`vmirtGetSMPPrevSibling`	`opProcessorSiblingPrevious`
`vmirtGetSMPNextSibling`	`opProcessorSiblingNext`
`vmirtGetSMPActiveSibling`	
`vmirtGetSMPCpuType`	
`vmirtGetSMPIndex`	`opProcessorIndex`
`vmirtSetSMPIndex`	
`vmirtIterAllChildren`	`opProcessorIterChildren`
`vmirtIterAllDescendants`	`opProcessorIterDescendants`
`vmirtIterAllProcessors`	`opProcessorIterAll`

For related OP functions, see section 3.6.

2.10 Object File Access

For related OP functions, see section 3.7.

2.10.1 Translation between Object File Addresses and Names

These functions handle translation between object file addresses and names:

```
vmirtAddressLookup
vmirtSymbolLookup
```

2.10.2 Add and Access Object Files

These functions are used to add and access object files:

```
vmirtAddSymbolFile
vmirtNextSymbolFile
vmirtGetSymbolFileName
```

2.10.3 Query Object File Symbols

These functions are used to query symbol information in object files:

```
vmirtGetSymbolByName
vmirtGetSymbolByNameFile
vmirtGetSymbolByAddr
vmirtGetSymbolByAddrFile
vmirtNextSymbolByName
vmirtNextSymbolByAddr
vmirtPrevSymbolByAddr
vmirtNextSymbolByNameFile
vmirtNextSymbolByAddrFile
vmirtPrevSymbolByAddrFile
vmirtGetSymbolName
vmirtGetSymbolAddr
vmirtGetSymbolLoadAddr
vmirtGetSymbolType
vmirtGetSymbolBinding
vmirtGetSymbolSize
```

2.10.4 Query Object File File/Line

These functions are used to query file/line information in object files:

```
vmirtGetFLByAddr
vmirtGetFLByAddrFile
vmirtNextFLByAddr
vmirtPrevFLByAddr
vmirtNextFLByAddrFile
vmirtPrevFLByAddrFile
vmirtGetFLFileName
vmirtGetFLLineNumber
vmirtGetFLAddr
```

2.11 Range Table Hash

These functions are used to implement address range hash tables:

```
vmirtNewRangeTable
vmirtFreeRangeTable
vmirtInsertRangeEntry
vmirtRemoveRangeEntry
vmirtGetFirstRangeEntry
vmirtGetNextRangeEntry
vmirtGetRangeEntryLow
vmirtGetRangeEntryHigh
vmirtGetRangeEntryUserData
vmirtSetRangeEntryUserData
```

2.12 Shared Data

These functions are used to access shared data:

`vmirtFindSharedData`	`opSharedDataFind`
`vmirtFindAddSharedData`	`opSharedDataFindAdd`
`vmirtFindProcessorSharedData`	
`vmirtFindAddProcessorSharedData`	
`vmirtRemoveSharedData`	`opSharedDataDelete`
`vmirtSetSharedDataValue`	`opSharedDataValueSet`
`vmirtGetSharedDataValue`	`opSharedDataValueSet`
`vmirtRegisterListener`	`opSharedDataListenerRegister`
`vmirtUnregisterListener`	`opSharedDataListenerUnregister`
`vmirtWriteListeners`	`opSharedDataListenersWrite`

For related OP functions, see section 3.8.

2.13 Notifiers

2.13.1 Register Update Notifiers

These functions are used to handle notifiers when registers are written:

```
vmirtAddRegisterWatchCallback
vmirtDeleteRegisterWatchCallback
```

2.13.2 Branch Reason Notifiers

These functions are used to handle notifiers when branch events occur:

```
vmirtRegisterBranchNotifier
vmirtUnregisterBranchNotifier
```

2.14 Save/Restore Support

These functions implement save and restore of processor state:

`vmirtSave`	`opStateItemSave`
`vmirtRestore`	`opStateItemRestore`
`vmirtSaveElement`	
`vmirtRestoreElement`	
`vmirtSaveModelTimer`	
`vmirtRestoreModelTimer`	
`vmirtSaveDomain`	`opMemoryStateSave`
`vmirtRestoreDomain`	`opMemoryStateRestore`
`vmirtGetPostSlotCB`	
`vmirtSetPostSlotCB`	

For related OP functions, see section 3.11.

2.15 Debug View Support

These functions implement debug view objects:

```
vmirtGetProcessorViewObject
vmirtSetViewObjectUserData
vmirtGetViewObjectUserData
vmirtAddViewObject
```

```
vmirtSetViewObjectConstValue
vmirtSetViewObjectRefValue
vmirtSetViewObjectValueCallback
vmirtAddViewAction
vmirtAddViewEvent
vmirtNextViewEvent
vmirtTriggerViewEvent
vmirtDeleteViewObject
```

2.16 Instruction Attributes

These are instruction attributes interface functions (some in `vmiInstructionAttrs.h`):

```
vmiiaGetAttrs                        opProcessorInstructionAttributes
vmiiaConvertRegInfo                  opRegConvert
vmirtRegImplRaw
vmirtEvaluateCondition
```

For related OP functions, see section 3.12.

2.17 Shared Object / Dynamic Linked Library Loading

These functions implement loading of shared objects:

```
vmirtDLOpen
vmirtDLError
vmirtDLSymbol
vmirtDLClose
```

2.18 Command Interpreter

These functions implement access to the standard command interpreter:

```
vmirtAddCommand
vmirtAddCommandParse
vmirtAddArg
vmirtAddArgEnum
vmirtFindArgValue
```

For related OP functions, see section 3.13.

2.19 Debugger Integration

These functions implement integration with debug:

```
vmirtGetProcessorScope
vmirtEvaluateGDBExpression
vmirtEvaluateCodeLocation
vmirtDisassemble                     opProcessorDisassemble
vmirtInstructionBytes                opProcessorInstructionBytes
```

For related OP functions, see section 3.14.

2.20 Trace Integration

These functions implement integration with trace:

```
vmirtTraceOnAfter                    opProcessorTraceOnAfter
```

`vmirtTraceOffAfter`	`opProcessorTraceOffAfter`

For related OP functions, see section 3.18.

2.21 Documentation

These functions implement processor documentation (in `vmiDoc.h`):

`vmidocAddSection`	`opDocSectionAdd`
`vmidocAddText`	`opDocTextAdd`
`vmidocProcessor`	
`vmidocAddFields`	
`vmidocAddField`	
`vmidocAddConstField`	

For related OP functions, see section 3.19.

2.22 Messages

These functions implement messaging and output (in `vmiMessage.h`):

`vmiMessage`	`opMessage`
`vmiVMessage`	`opVMessage`
`vmiPrintf`	`opPrintf`
`vmiVPrintf`	`opVPrintf`
`vmiAbort`	

For related OP functions, see section 3.20.

2.23 HTTP Interface

This function implements the HTTP interface (in `vmiHTTP.h`):

`vmihttpOpen`	`opModuleHTTPOpen`

For related OP functions, see section 3.21.

2.24 Licensing

These functions are used to implement licensing:

```
vmirtGetLicense
vmirtGetLicenseErrString
```

2.25 Encapsulated Model

These functions implement encapsulated model semihosting:

```
vmirtEncapIntercept
```

3 OP Functions

3.1 Simulation

3.1.1 Session Control

These functions control the simulation session:

```
opSessionAtExit
opSessionBuildDate
opSessionCancelTextRedirect
opSessionDebuggerNotifiersAdd
opSessionDestFnSet
opSessionExit
opSessionFeaturesSet
opSessionInit
opSessionProductName
opSessionProductVersion
opSessionTerminate
opSessionTextRedirect
```

3.1.2 Execution Control

These functions control simulation flow:

```
opProcessorSimulate
opRootModuleSetSimulationRandomSeed
opRootModuleSetSimulationStopTime
opRootModuleSetSimulationTimePrecision
opRootModuleSetSimulationTimeSlice
opRootModuleSetWallClockFactor
opRootModuleSimulate
opRootModuleStopReason
opRootModulePostElaborate
opRootModulePostSimulate
opRootModulePreSimulate
opRootModuleSetDebugStopTime
opRootModuleTimeAdvance
opObjectSimulatorPhase
opObjectSimulatorPhaseString
opStopReasonString
```

3.1.3 Environment Access

These functions are used to access simulation environment features:

```
opBanner
opErrors
opLicPersonalitySet
opNoBanner
opProductSet
```

3.1.4 Module Simulation

These functions control execution of a module:

opModuleFinish	
opModuleFinishStatus	

3.2 *Program Counter and Code Dictionary*

These functions handle access to simulated program counter and invalidation of code dictionary:

opMemoryFlush	
opMemoryNativeFlush	
opProcessorFlush	
opProcessorPC	vmirtGetPC
opProcessorPCDS	vmirtGetPCDS
opProcessorPCInDS	
opProcessorPCNext	
opProcessorPCSet	vmirtSetPC

For related VMI functions, see section 2.3.

3.3 *Time and Cycle Counts*

3.3.1 Instruction/Cycle Counting and Interrupt

These functions handle instruction and cycle counts:

opProcessorClocks	
opProcessorCycleCount	vmirtGetICount
opProcessorICount	vmirtGetExecutedICount

For related VMI functions, see section 2.5.1.

3.3.2 Simulated Time

These functions handle interaction with simulated time:

opProcessorClocksUntilTime	
opProcessorTime	vmirtGetLocalTime
opEventTimeNext	
opModuleCurrentTime	

For related VMI functions, see section 2.5.2.

3.3.3 Delay Estimation

These functions handle delay estimation:

opProcessorDelay	
opProcessorDelayAdd	
opProcessorDerate	vmirtSetDerateFactor
opProcessorSkipCyclesAdd	vmirtAddSkipCount
opProcessorSkipCycles	vmirtGetSkipCount

For related VMI functions, see section 2.5.3.

3.4 Platform Component Creation, Connection and Query

3.4.1 Bus Bridges

These functions operate on bus bridges:

```
opBridgeBusConnect
opBridgeNew
opBridgeNext
```

3.4.2 Buses

These functions operate on buses:

```
opBusAddrBits
opBusMappedRangeNext
opBusMaxAddress
opBusNew
opBusNext
opBusShow
```

3.4.3 Bus Slaves

These functions operate on bus slaves:

```
opBusSlaveAddrHi
opBusSlaveAddrLo
opBusSlaveNew
opBusSlaveNext
```

3.4.4 Bus Ports

These functions operate on bus ports:

```
opBusPortAddrBitsDefault
opBusPortAddrBitsMax
opBusPortAddrBitsMin
opBusPortAddrHi
opBusPortDescription
opBusPortDomainType
opBusPortDomainTypeString
opBusPortIsDynamic
opBusPortMMRegisterNext
opBusPortMustConnect
opBusPortType
opBusPortTypeString
```

For related VMI functions, see section 2.6.2.

3.4.5 Bus Port Connections

These functions operate on bus port connections:

```
opBusPortConnAddrHi
opBusPortConnAddrLo
opBusPortConnBus
opBusPortConnIsDynamic
opBusPortConnMapNotify
opBusPortConnNext
opBusPortConnType
```

```
opBusPortConnTypeString
```

3.4.6 Extensions

These functions operate on processor extensions:

```
opExtElabExtension
opExtensionNew
opExtensionPath
```

3.4.7 FIFOs

These functions operate on FIFOs:

```
opFIFODepth
opFIFONew
opFIFONext
opFIFOShow
```

3.4.8 FIFO Ports

These functions operate on FIFO ports:

```
opFIFOPortDescription
opFIFOPortMustConnect
opFIFOPortType
opFIFOPortTypeString
opFIFOPortWidth
```

For related VMI functions, see section 2.6.4.

3.4.9 FIFO Port Connections

These functions operate on FIFO port connections:

```
opFIFOPortConnFIFO
opFIFOPortConnNext
opFIFOPortConnWidth
```

3.4.10 MMCs

These functions operate on memory model components:

```
opMMCBusConnect
opMMCNew
opMMCNext
opMMCPath
opMMCTransparent
```

3.4.11 Memory-Mapped Registers

These functions operate on memory-mapped registers:

```
opMMRegisterBits
opMMRegisterDescription
opMMRegisterFieldBits
opMMRegisterFieldDescription
opMMRegisterFieldNext
opMMRegisterFieldOffset
opMMRegisterFieldReadable
```

```
opMMRegisterFieldReset
opMMRegisterFieldWritable
opMMRegisterIndex
opMMRegisterIsVolatile
opMMRegisterName
opMMRegisterOffset
opMMRegisterReadable
opMMRegisterView
opMMRegisterWritable
```

3.4.12 Memories

These functions operate on memories:

```
opMemoryBusConnect
opMemoryMaxAddress
opMemoryNativeNew
opMemoryNew
opMemoryNext
opMemorySpecParse
```

3.4.13 Modules

These functions operate on modules:

```
opFixedModuleNew
opModuleBusShow
opModuleNew
opModuleNewFromAttrs
opModuleNext
opModuleObject
opModulePath
opModulePurpose
opModuleShow
opRoot
opRootModuleDelete
opRootModuleNew
```

3.4.14 Nets

These functions operate on nets:

```
opNetNew
opNetNext
opNetShow
opNetValue
opNetValuePrevious
opNetWrite
opNetMonitorNext
opNetWriteMonitorAdd
```

3.4.15 Net Ports

These functions operate on net ports:

```
opNetPortDescription
opNetPortMustConnect
opNetPortType
opNetPortTypeString
```

For related VMI functions, see section 2.6.3.

3.4.16 Net Port Connections

These functions operate on net port connections:

```
opNetPortConnNet
opNetPortConnNext
opNetPortConnType
```

3.4.17 Objects

These functions operate on generic objects:

```
opObjectBusPortConnNext
opObjectBusPortNext
opObjectByName
opObjectClass
opObjectClassSet
opObjectExtElabNext
opObjectExtensionNext
opObjectFIFOPortConnNext
opObjectFIFOPortNext
opObjectHierName
opObjectModule
opObjectName
opObjectNetConnect
opObjectNetPortConnNext
opObjectNetPortNext
opObjectPacketnetPortConnNext
opObjectPacketnetPortNext
opObjectParent
opObjectReleaseStatus
opObjectReleaseStatusString
opObjectRootModule
opObjectType
opObjectVLNV
opObjectVisibility
opObjectVisibilityString
opVoidParent
opVoidByName
```

3.4.18 Packet Nets

These functions operate on packet nets:

```
opPacketnetMaxBytes
opPacketnetNew
opPacketnetNext
opPacketnetShow
opPacketnetWrite
opPacketnetMonitorNext
opPacketnetWriteMonitorAdd
```

3.4.19 Packet Net Port Connections

These functions operate on packet net port connections:

```
opPacketnetPortConnNext
opPacketnetPortConnPacketnet
opPacketnetPortDescription
opPacketnetPortMustConnect
```

3.4.20 Peripherals

These functions operate on peripherals:

```
opPeripheralBusConnectMaster
opPeripheralBusConnectSlave
opPeripheralBusConnectSlaveDynamic
opPeripheralExtensionNew
opPeripheralFIFOConnect
opPeripheralNew
opPeripheralNext
opPeripheralPacketnetConnect
opPeripheralPath
opPeripheralSerialNotify
opPeripheralStopReason
```

3.4.21 Processors

These functions operate on processors:

opProcessorAMP	
opProcessorAlternateName	
opProcessorBusConnect	
opProcessorBusConnectMaster	
opProcessorBusConnectSlave	
opProcessorDefaultSemihost	
opProcessorDescription	
opProcessorElfCodes	
opProcessorEndian	
opProcessorExceptionCurrent	
opProcessorExceptionNext	
opProcessorExtensionNew	
opProcessorFIFOConnect	
opProcessorFamily	
opProcessorFaultAddress	
opProcessorGroupH	
opProcessorGroupL	
opProcessorHelper	
opProcessorLoadPhysical	
opProcessorModeCurrent	vmirtGetCurrentMode
opProcessorModeNext	vmirtGetNextMode
opProcessorNew	
opProcessorNewFromAttrs	
opProcessorNewWithSemihost	
opProcessorNext	
opProcessorPath	
opProcessorQLQualified	
opProcessorStopReason	
opProcessorCurrent	vmirtGetCurrentProcessor
opProcessorVariant	vmirtProcessorVariant
opProcessorExit	vmirtExit
opProcessorFinish	vmirtFinish
opProcessorFreeze	
opProcessorFrozen	
opProcessorHalt	vmirtHalt
opProcessorUnfreeze	
opProcessorYield	vmirtYieldControl
opInterrupt	vmirtInterrupt
opInterruptRSP	vmirtStop

For related VMI functions, see section 2.2.

3.4.22 VLNV

These functions operate on VLNV entries:

```
opVLNVIter
opVLNVLibrary
opVLNVName
opVLNVNew
opVLNVOld
opVLNVString
opVLNVVendor
opVLNVVersion
```

3.5 *Memory Operations*

3.5.1 Generic Load/Store

These functions implement load and store operations:

`opBusRead`	
`opBusWrite`	
`opMemoryRead`	`vmirtReadNByteDomain`
`opMemoryWrite`	`vmirtWriteNByteDomain`
`opProcessorRead`	
`opProcessorReadAbort`	
`opProcessorWrite`	
`opProcessorWriteAbort`	

For related VMI functions, see section 2.7.1.

3.5.2 Memory Callbacks

These functions handle installation and removal of callback functions on memory accesses:

`opBusFetchMonitorAdd`	
`opBusFetchMonitorDelete`	
`opBusReadMonitorAdd`	
`opBusReadMonitorDelete`	
`opBusWriteMonitorAdd`	
`opBusWriteMonitorDelete`	
`opMemoryFetchMonitorAdd`	`vmirtAddFetchCallback`
`opMemoryFetchMonitorDelete`	`vmirtRemoveFetchCallback`
`opMemoryReadMonitorAdd`	`vmirtAddReadCallback`
`opMemoryReadMonitorDelete`	`vmirtRemoveReadCallback`
`opMemoryWriteMonitorAdd`	`vmirtAddWriteCallback`
`opMemoryWriteMonitorDelete`	`vmirtRemoveWriteCallback`
`opProcessorFetchMonitorAdd`	
`opProcessorFetchMonitorDelete`	
`opProcessorReadMonitorAdd`	
`opProcessorReadMonitorDelete`	
`opProcessorWriteMonitorAdd`	
`opProcessorWriteMonitorDelete`	

For related VMI functions, see section 2.7.2.

3.5.3 Memory Manipulation

These functions are used to query and manipulate `memDomain` objects:

`opDynamicBridge`	`vmirtAliasMemory`
`opDynamicUnbridge`	`vmirtUnaliasMemory`
`opMemoryNativeDynamic`	`vmirtMapNativeMemory`
`opBusPrivSet`	`vmirtProtectMemory`
`opBusSlavePriv`	
`opMemoryPriv`	
`opModuleDomainDebug`	`vmirtDebugDomain`

For related VMI functions, see section 2.7.3.

3.6 SMP Processor Hierarchy

These functions are used to traverse SMP processor hierarchy:

`opProcessorChild`	`vmirtGetSMPChild`
`opProcessorIndex`	`vmirtGetSMPIndex`
`opProcessorIsLeaf`	
`opProcessorIterAll`	`vmirtIterAllProcessors`
`opProcessorIterChildren`	`vmirtIterAllChildren`
`opProcessorIterDescendants`	`vmirtIterAllDescendants`
`opProcessorParent`	`vmirtGetSMPParent`
`opProcessorSiblingNext`	`vmirtGetSMPNextSibling`
`opProcessorSiblingPrevious`	`vmirtGetSMPPrevSibling`

For related VMI functions, see section 2.9.

3.7 Application File Access

These functions are used to access application files:

```
opApplicationControls
opApplicationElfCode
opApplicationEndian
opApplicationEntry
opApplicationHeaderRead
opApplicationLoaderInstall
opApplicationOffset
opApplicationPath
opBusApplicationLoad
opMemoryApplicationLoad
opObjectApplicationNext
opProcessorApplicationLoad
opProcessorApplicationRead
opProcessorApplicationSymbolAdd
```

For related VMI functions, see section 2.10.

3.8 Shared Data

These functions are used to access shared data:

`opSharedDataDelete`	`vmirtRemoveSharedData`
`opSharedDataFind`	`vmirtFindSharedData`
`opSharedDataFindAdd`	`vmirtFindAddSharedData`
`opSharedDataListenerRegister`	`vmirtRegisterListener`
`opSharedDataListenerUnregister`	`vmirtUnregisterListener`
`opSharedDataListenersWrite`	`vmirtWriteListeners`
`opSharedDataValueGet`	`vmirtSetSharedDataValue`

opSharedDataValueSet	vmirtSetSharedDataValue

For related VMI functions, see section 2.12.

3.9 *Processor Registers, Exceptions and Modes*

3.9.1 Processor Registers

These functions are used to access processor registers:

opProcessorRegByIndex	
opProcessorRegByName	vmirtGetRegByName
opProcessorRegByUsage	
opProcessorRegDump	
opProcessorRegGroupByName	vmirtGetRegGroupByName
opProcessorRegGroupNext	vmirtGetNextRegGroup
opProcessorRegIsExtension	
opProcessorRegNext	vmirtGetNextReg
opProcessorRegNextGPacket	vmirtGetNextReg
opProcessorRegNextPPacket	vmirtGetNextReg
opProcessorRegRead	vmirtRegRead
opProcessorRegReadByName	
opProcessorRegWrite	vmirtRegRead
opProcessorRegWriteByName	
opRegAccessEnum	
opRegAccessString	
opRegBits	
opRegDescription	
opRegGroup	
opRegGroupName	
opRegGroupRegNext	vmirtGetNextRegInGroup
opRegIndex	
opRegIsAlias	
opRegName	
opRegReadOnly	
opRegUsageEnum	
opRegUsageString	

For related VMI functions, see section 2.6.1.

3.9.2 Mode Access

These functions implement access to processor modes:

```
opModeCode
opModeDescription
opModeName
```

3.9.3 Exception Access

These functions implement access to processor exceptions:

```
opExceptionCode
opExceptionDescription
opExceptionName
```

For related VMI functions, see section 2.6.5.

3.10 Parameters

These functions implement object parameters.

3.10.1 Formal Parameters

```
opFormalBoolDefaultValue
opFormalDescription
opFormalEnumDefault
opFormalEnumDescription
opFormalEnumNext
opFormalEnumValue
opFormalGroup
opFormalGroupDescription
opFormalGroupName
opFormalInt32Limits
opFormalInt64Limits
opFormalStringDefaultValue
opFormalStringMaxLength
opFormalSystem
opFormalType
opFormalTypeString
opFormalUns32Limits
opFormalUns64Limits
opFormalValueOrigin
opFormalValueOriginString
opFormaldoubleLimits
opModuleFormalsShow
opObjectFormalGroupNext
opObjectFormalNext
```

3.10.2 Actual Parameters

```
opObjectParamBoolValue
opObjectParamDoubleValue
opObjectParamEndianValue
opObjectParamEnumValue
opObjectParamInt32Value
opObjectParamInt64Value
opObjectParamNext
opObjectParamPtrValue
opObjectParamStringValue
opObjectParamUns32Value
opObjectParamUns64Value
opParamBoolOverride
opParamBoolSet
opParamDoubleOverride
opParamDoubleSet
opParamEndianOverride
opParamEndianSet
opParamEnumOverride
opParamEnumSet
opParamInt32Override
opParamInt32Set
opParamInt64Override
opParamInt64Set
opParamListFromArray
opParamPtrOverride
opParamPtrSet
opParamPtrValue
opParamStringOverride
opParamStringSet
opParamType
```

```
opParamTypeString
opParamUns32Override
opParamUns32Set
opParamUns64Override
opParamUns64Set
```

3.11 Save/Restore Support

These functions implement save and restore of processor state:

opFIFOStateRestore	
opFIFOStateRestoreFile	
opFIFOStateSave	
opFIFOStateSaveFile	
opMMCStateRestore	
opMMCStateRestoreFile	
opMMCStateSave	
opMMCStateSaveFile	
opMemoryStateRestore	vmirtRestoreDomain
opMemoryStateRestoreFile	
opMemoryStateSave	vmirtSaveDomain
opMemoryStateSaveFile	
opNetStateRestore	
opNetStateRestoreFile	
opNetStateSave	
opNetStateSaveFile	
opObjectSaveRestoreSupported	
opProcessorStateRestore	
opProcessorStateRestoreFile	
opProcessorStateSave	
opProcessorStateSaveFile	
opRootModuleStateRestore	
opRootModuleStateRestoreFile	
opRootModuleStateSave	
opRootModuleStateSaveFile	
opStateItemRestore	vmirtRestore
opStateItemSave	vmirtSave

For related VMI functions, see section 2.14.

3.12 Instruction Attributes

These are instruction attributes interface functions:

opProcessorInstructionAttributes	vmiiaGetAttrs
opRegConvert	vmiiaConvertRegInfo

For related VMI functions, see section 2.16.

3.13 Command Interpreter

These functions implement access to the standard command interpreter:

```
opCmdArgUsed
opCmdDefaultApplication
opCmdErrorHandler
opCmdParseArgs
opCmdParseFile
opCmdParseStd
```

```
opCmdParserAdd
opCmdParserDelete
opCmdParserNew
opCmdParserReplace
opCmdUsageMessage
opCommandArgDescription
opCommandArgIterAll
opCommandArgName
opCommandArgType
opCommandArgTypeString
opCommandCall
opCommandCallByName
opCommandHelp
opCommandIterAll
opCommandStringCall
opModuleCommandsShow
opObjectCommandNext
opProcessorCommandIterAll
```

For related VMI functions, see section 2.18.

3.14 Debugger Integration

These functions implement integration with debuggers such as gdb:

```
opProcessorDebug
opProcessorDebugHelper
opProcessorDisassemble                    vmirtDisassemble
opProcessorGdbFlags
opProcessorGdbPath
opProcessorInstructionBytes               vmirtInstructionBytes
```

For related VMI functions, see section 2.19.

3.15 Breakpoints

These functions implement support for breakpoints:

```
opProcessorBreakpointAddrClear
opProcessorBreakpointAddrSet
opProcessorBreakpointICountClear
opProcessorBreakpointICountSet
```

3.16 Watchpoints

These functions implement support for watchpoints:

```
opBusAccessWatchpointNew
opBusReadWatchpointNew
opBusWriteWatchpointNew
opMemoryAccessWatchpointNew
opMemoryReadWatchpointNew
opMemoryWriteWatchpointNew
opProcessorAccessWatchpointNew
opProcessorExceptionWatchpointNew
opProcessorModeWatchpointNew
opProcessorReadWatchpointNew
opProcessorRegWatchpointNew
opProcessorWriteWatchpointNew
opRootModuleWatchpointNext
```

opWatchpointAddressHi	
opWatchpointAddressLo	
opWatchpointDelete	
opWatchpointReg	
opWatchpointRegCurrentValue	
opWatchpointRegPreviousValue	
opWatchpointReset	
opWatchpointTriggeredBy	
opWatchpointType	
opWatchpointUserData	

3.17 Triggers

These functions implement triggers on module and stop reason events:

opModuleTriggerAdd	
opModuleTriggerDelete	
opProcessorStopHandlerAdd	
opProcessorStopHandlerDelete	

3.18 Trace Integration

These functions implement integration with trace:

opPrintfTrace	
opProcessorTraceBufferDisable	
opProcessorTraceBufferDump	
opProcessorTraceBufferEnable	
opProcessorTraceHighPCSet	
opProcessorTraceLowPCSet	
opProcessorTraceOffAfter	vmirtTraceOffAfter
opProcessorTraceOnAfter	vmirtTraceOnAfter

For related VMI functions, see section 2.20.

3.19 Documentation

These functions implement documentation generation:

opDocChildNext	
opDocFieldOffset	
opDocFieldWidth	
opDocIsTitle	
opDocNodeType	
opDocSectionAdd	vmidocAddSection
opDocText	
opDocTextAdd	vmidocAddText
opModuleDocSectionAdd	
opObjectDocNodeNext	

For related VMI functions, see section 2.21.

3.20 Messages

These functions implement messaging and output:

opLastMessage	
opMessage	vmiMessage
opMessageDisable	

opMessageEnable	
opMessageQuiet	
opMessageSetNoWarn	
opMessageSetQuiet	
opMessageVerbose	
opModuleDiagnosticLevelSet	
opPeripheralDiagnosticLevelSet	
opPrintf	vmiPrintf
opResetErrors	
opSprintf	
opVAbort	
opVMessage	vmiVMessage
opVPrintf	vmiVPrintf
opVSprintf	

For related VMI functions, see section 2.22.

3.21 HTTP Interface

This function implements the HTTP interface:

opModuleHTTPOpen	vmihttpOpen

For related VMI functions, see section 2.23.