NAS VAAI Development Kit - ESX Server 7.0.1

API Reference vmkapi_2_7_0_0

Generated by Doxygen 1.5.8

Fri Sep 4 11:40:47 2020

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 VixDiskLibBlock Struct Reference

Data Fields

- VixDiskLibSectorType offset
- VixDiskLibSectorType length

3.1.1 Field Documentation

3.1.1.1 VixDiskLibSectorType VixDiskLibBlock::offset

3.1.1.2 VixDiskLibSectorType VixDiskLibBlock::length

3.2 VixDiskLibBlockList Struct Reference

Data Fields

- uint32 numBlocks
- VixDiskLibBlock blocks [1]

3.2.1 Field Documentation

3.2.1.1 uint32 VixDiskLibBlockList::numBlocks

3.2.1.2 VixDiskLibBlock VixDiskLibBlockList::blocks[1]

3.3 VixDiskLibConnectParams Struct Reference

Data Structures

- union VixDiskLibCreds

Data Fields

- char ∗ vmxSpec
- char ∗ serverName
- char ∗ thumbPrint
- long privateUse
- VixDiskLibCredType credType
- union VixDiskLibConnectParams::VixDiskLibCreds creds
- uint32 port
- uint32 nfcHostPort
- char ∗ vimApiVer
- char reserved [8]
- VixDiskLibConnectParamsState ∗ state
- union {
  VixDiskLibVStorageObjectSpec vStorageObjSpec
  VixDiskLibDatastoreSpec dsSpec
  } spec
- VixDiskLibSpecType specType

3.3.1 Detailed Description

VixDiskLibConnectParams - Connection setup parameters.

vmxSpec is required for opening a virtual disk on a datastore through the Virtual Center or ESX server. vmxSpec is of the form: <vmxPathName>?dcPath=<dcpath>&dsName=<dsname> where vmxPathName is the fullpath for the VMX file, dcpath is the inventory path of the datacenter and dsname is the datastore name.

Inventory path for the datacenter can be read off the Virtual Center client's inventory tree.

Example VM spec: "MyVm/MyVm.vmx?dcPath=Path/to/MyDatacenter&dsName=storage1"

3.3.2 Field Documentation

3.3.2.1 char∗ VixDiskLibConnectParams::vmxSpec

3.3.2.2 char∗ VixDiskLibConnectParams::serverName

3.3.2.3 char∗ VixDiskLibConnectParams::thumbPrint

3.3.2.4 long VixDiskLibConnectParams::privateUse

3.3.2.5 VixDiskLibCredType VixDiskLibConnectParams::credType

3.3.2.6 union VixDiskLibConnectParams::VixDiskLibCreds VixDiskLibConnectParams::creds

3.3.2.7 uint32 VixDiskLibConnectParams::port

3.3.2.8 uint32 VixDiskLibConnectParams::nfcHostPort

3.3.2.9 char∗ VixDiskLibConnectParams::vimApiVer

3.3.2.10 char VixDiskLibConnectParams::reserved[8]

3.3.2.11 VixDiskLibConnectParamsState∗ VixDiskLibConnectParams::state

3.3.2.12 VixDiskLibVStorageObjectSpec VixDiskLibConnectParams::vStorageObjSpec

3.3.2.13 VixDiskLibDatastoreSpec VixDiskLibConnectParams::dsSpec

3.3.2.14 union { ... } VixDiskLibConnectParams::spec

3.3.2.15 VixDiskLibSpecType VixDiskLibConnectParams::specType

3.4 VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds Union Reference

Data Structures

- struct VixDiskLibSessionIdCreds
- struct VixDiskLibUidPasswdCreds

Data Fields

- struct VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibUidPasswdCreds uid
- struct VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibSessionIdCreds sessionId
- struct VixDiskLibTicketIdCreds ∗ ticketId

3.4.1 Field Documentation

3.4.1.1 struct VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibUidPasswdCreds VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::uid

3.4.1.2 struct VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibSessionIdCreds VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::sessionId

3.4.1.3 struct VixDiskLibTicketIdCreds∗ VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::ticketId

`[read]`

3.5 VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::V Struct Reference

Data Fields

- char ∗ cookie
- char ∗ userName
- char ∗ key

3.5.1 Field Documentation

3.5.1.1 char∗ VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskLibCreds::VixD

3.5.1.2 char∗ VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskLibCreds::VixD

3.5.1.3 char∗ VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskLibCreds::VixD

3.6 VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds:: Struct Reference

Data Fields

- char ∗ userName
- char ∗ password

3.6.1 Field Documentation

3.6.1.1 char∗ VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskLibCreds::VixD

3.6.1.2 char∗ VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskLibCreds::VixD

3.7 VixDiskLibCreateParams Struct Reference

Data Fields

- VixDiskLibDiskType diskType
- VixDiskLibAdapterType adapterType
- uint16 hwVersion
- VixDiskLibSectorType capacity
- uint32 logicalSectorSize
- uint32 physicalSectorSize

3.7.1 Field Documentation

3.7.1.1 VixDiskLibDiskType VixDiskLibCreateParams::diskType

3.7.1.2 VixDiskLibAdapterType VixDiskLibCreateParams::adapterType

3.7.1.3 uint16 VixDiskLibCreateParams::hwVersion

3.7.1.4 VixDiskLibSectorType VixDiskLibCreateParams::capacity

3.7.1.5 uint32 VixDiskLibCreateParams::logicalSectorSize

3.7.1.6 uint32 VixDiskLibCreateParams::physicalSectorSize

3.8 VixDiskLibDatastoreSpec Struct Reference

Data Fields

- char ∗ datastoreMoRef

3.8.1 Field Documentation

3.8.1.1 char∗ VixDiskLibDatastoreSpec::datastoreMoRef

3.9 VixDiskLibGeometry Struct Reference

Data Fields

- uint32 cylinders
- uint32 heads
- uint32 sectors

3.9.1 Field Documentation

3.9.1.1 uint32 VixDiskLibGeometry::cylinders

3.9.1.2 uint32 VixDiskLibGeometry::heads

3.9.1.3 uint32 VixDiskLibGeometry::sectors

3.10 VixDiskLibInfo Struct Reference

Data Fields

- VixDiskLibGeometry biosGeo
- VixDiskLibGeometry physGeo
- VixDiskLibSectorType capacity
- VixDiskLibAdapterType adapterType
- int numLinks
- char ∗ parentFileNameHint
- char ∗ uuid
- uint32 logicalSectorSize
- uint32 physicalSectorSize

3.10.1 Field Documentation

3.10.1.1 VixDiskLibGeometry VixDiskLibInfo::biosGeo

3.10.1.2 VixDiskLibGeometry VixDiskLibInfo::physGeo

3.10.1.3 VixDiskLibSectorType VixDiskLibInfo::capacity

3.10.1.4 VixDiskLibAdapterType VixDiskLibInfo::adapterType

3.10.1.5 int VixDiskLibInfo::numLinks

3.10.1.6 char∗ VixDiskLibInfo::parentFileNameHint

3.10.1.7 char∗ VixDiskLibInfo::uuid

3.10.1.8 uint32 VixDiskLibInfo::logicalSectorSize

3.10.1.9 uint32 VixDiskLibInfo::physicalSectorSize

3.11 VixDiskLibNasPlugin Struct Reference

Data Fields

- VixDiskLibPlugin diskLibPlugin
- VixDiskLibNasPluginStartSession ∗ StartSession
- VixDiskLibNasPluginEndSession ∗ EndSession
- VixDiskLibNasPluginExecutePrimitive ∗ ExecPrimitive
- VixDiskLibNasPluginSupportStatus ∗ SupportStatus

3.11.1 Detailed Description

Plugin function table to be exported by a NAS plugin. See the description of the various data types for explanation.

3.11.2 Field Documentation

3.11.2.1 VixDiskLibPlugin VixDiskLibNasPlugin::diskLibPlugin

3.11.2.2 VixDiskLibNasPluginStartSession∗ VixDiskLibNasPlugin::StartSession

3.11.2.3 VixDiskLibNasPluginEndSession∗ VixDiskLibNasPlugin::EndSession

3.11.2.4 VixDiskLibNasPluginExecutePrimitive∗ VixDiskLibNasPlugin::ExecPrimitive

3.11.2.5 VixDiskLibNasPluginSupportStatus∗ VixDiskLibNasPlugin::SupportStatus

3.12 VixDiskLibNasPluginCloneFileParams Struct Reference

Data Fields

- VixDiskLibNasPluginCommonParams common
- char ∗ srcFileName
- char ∗ dstFileName
- VixDiskLibNasPluginCloneFileFlags cloneFlags
- VixDiskLibNasPluginDataStoreParams ∗ srcDataStoreInfo

3.12.1 Field Documentation

3.12.1.1 VixDiskLibNasPluginCommonParams VixDiskLibNasPluginCloneFileParams::common

3.12.1.2 char∗ VixDiskLibNasPluginCloneFileParams::srcFileName

3.12.1.3 char∗ VixDiskLibNasPluginCloneFileParams::dstFileName

3.12.1.4 VixDiskLibNasPluginCloneFileFlags VixDiskLibNasPluginCloneFileParams::cloneFlags

3.12.1.5 VixDiskLibNasPluginDataStoreParams∗ VixDiskLibNasPluginCloneFileParams::srcDataStoreInfo

3.13 VixDiskLibNasPluginCommonParams Struct Reference

Data Fields

- VixDiskLibNasPluginPrimitiveID primitiveID
- VixDiskLibNasPluginProgressRecord * progressRecord
- VixDiskLibNasPluginResultCommon * result

3.13.1 Field Documentation

3.13.1.1 VixDiskLibNasPluginPrimitiveID VixDiskLibNasPluginCommonParams::primitiveID

3.13.1.2 VixDiskLibNasPluginProgressRecord* VixDiskLibNasPluginCommonParams::progressRecord

3.13.1.3 VixDiskLibNasPluginResultCommon* VixDiskLibNasPluginCommonParams::result

3.14 VixDiskLibNasPluginDataStoreParams Struct Reference

Data Fields

- char ∗ fsType
- uint32 fsVersion
- char ∗ remoteIP
- char ∗ remoteMountPoint
- char ∗ localMountPoint

3.14.1 Field Documentation

3.14.1.1 char∗ VixDiskLibNasPluginDataStoreParams::fsType

3.14.1.2 uint32 VixDiskLibNasPluginDataStoreParams::fsVersion

3.14.1.3 char∗ VixDiskLibNasPluginDataStoreParams::remoteIP

3.14.1.4 char∗ VixDiskLibNasPluginDataStoreParams::remoteMountPoint

3.14.1.5 char∗ VixDiskLibNasPluginDataStoreParams::localMountPoint

3.15 VixDiskLibNasPluginProgressRecord Struct Reference

Data Fields

- void ∗ private
- uint64 updateBytes
- uint64 progressBytes
- VixDiskLibNasPluginPeriodicCallback ∗ callback

3.15.1 Field Documentation

3.15.1.1 void∗ VixDiskLibNasPluginProgressRecord::private

3.15.1.2 uint64 VixDiskLibNasPluginProgressRecord::updateBytes

3.15.1.3 uint64 VixDiskLibNasPluginProgressRecord::progressBytes

3.15.1.4 VixDiskLibNasPluginPeriodicCallback∗ VixDiskLibNasPluginProgressRecord::callback

3.16 VixDiskLibNasPluginResultCommon Struct Reference

Data Fields

- VixError status

3.16.1 Field Documentation

3.16.1.1 VixError VixDiskLibNasPluginResultCommon::status

3.17 VixDiskLibNasPluginResvSpaceParams Struct Reference

Data Fields

- VixDiskLibNasPluginCommonParams common
- char ∗ fileName

3.17.1 Field Documentation

3.17.1.1 VixDiskLibNasPluginCommonParams VixDiskLibNasPluginResvSpaceParams::common

3.17.1.2 char∗ VixDiskLibNasPluginResvSpaceParams::fileName

3.18 VixDiskLibNasPluginSessionParams Struct Reference

Data Fields

- uint64 timeoutMS

3.18.1 Field Documentation

3.18.1.1 uint64 VixDiskLibNasPluginSessionParams::timeoutMS

3.19 VixDiskLibNasPluginStatXParams Struct Reference

Data Fields

- VixDiskLibNasPluginCommonParams common
- char ∗ fileName

3.19.1 Field Documentation

3.19.1.1 VixDiskLibNasPluginCommonParams VixDiskLibNasPluginStatXParams::common

3.19.1.2 char∗ VixDiskLibNasPluginStatXParams::fileName

3.20 VixDiskLibNasPluginStatXResult Struct Reference

Data Fields

- VixDiskLibNasPluginResultCommon common
- uint64 totalBytes
- uint64 allocedBytes
- uint64 uniqueBytes
- VixDiskLibNasPluginAllocType allocType

3.20.1 Field Documentation

3.20.1.1 VixDiskLibNasPluginResultCommon VixDiskLibNasPluginStatXResult::common

3.20.1.2 uint64 VixDiskLibNasPluginStatXResult::totalBytes

3.20.1.3 uint64 VixDiskLibNasPluginStatXResult::allocedBytes

3.20.1.4 uint64 VixDiskLibNasPluginStatXResult::uniqueBytes

3.20.1.5 VixDiskLibNasPluginAllocType VixDiskLibNasPluginStatXResult::allocType

3.21 VixDiskLibPlugin Struct Reference

Data Fields

- int majorVersion
- int minorVersion
- VixDiskLibPluginType type
- const char ∗ name
- VixDiskLibPluginInit ∗ Init
- VixDiskLibPluginDone ∗ Done

3.21.1 Detailed Description

Plugin function table to be exported for each plugin the plugin library contains. See the description of the various data types for explanation.

3.21.2 Field Documentation

3.21.2.1 int VixDiskLibPlugin::majorVersion

3.21.2.2 int VixDiskLibPlugin::minorVersion

Major version supported by this plugin.

3.21.2.3 VixDiskLibPluginType VixDiskLibPlugin::type

Minor version supported by this plugin.

3.21.2.4 const char∗ VixDiskLibPlugin::name

The type of plugin this is

3.21.2.5 VixDiskLibPluginInit∗ VixDiskLibPlugin::Init

Name associated with this plugin.

3.21.2.6 VixDiskLibPluginDone∗ VixDiskLibPlugin::Done

Optional

3.22 VixDiskLibSpec Union Reference

Data Fields

- char ∗ vmxSpec
- VixDiskLibVStorageObjectSpec vStorageObjSpec
- VixDiskLibDatastoreSpec dsSpec

3.22.1 Field Documentation

3.22.1.1 char∗ VixDiskLibSpec::vmxSpec

3.22.1.2 VixDiskLibVStorageObjectSpec VixDiskLibSpec::vStorageObjSpec

3.22.1.3 VixDiskLibDatastoreSpec VixDiskLibSpec::dsSpec

3.23 VixDiskLibVStorageObjectSpec Struct Reference

Data Fields

- char ∗ id
- char ∗ datastoreMoRef
- char ∗ ssId

3.23.1 Field Documentation

3.23.1.1 char∗ VixDiskLibVStorageObjectSpec::id

3.23.1.2 char∗ VixDiskLibVStorageObjectSpec::datastoreMoRef

3.23.1.3 char∗ VixDiskLibVStorageObjectSpec::ssId

3.24 VMPoint Struct Reference

Data Fields

- int x
- int y

3.24.1 Field Documentation

3.24.1.1 int VMPoint::x

3.24.1.2 int VMPoint::y

3.25 VMRect Struct Reference

Data Fields

- int left
- int top
- int right
- int bottom

3.25.1 Field Documentation

3.25.1.1 int VMRect::left

3.25.1.2 int VMRect::top

3.25.1.3 int VMRect::right

3.25.1.4 int VMRect::bottom

Chapter 4

File Documentation

4.1 distribute_vixDiskLib.h File Reference

Data Structures

- struct VixDiskLibGeometry
- struct VixDiskLibCreateParams
- struct VixDiskLibVStorageObjectSpec
- struct VixDiskLibDatastoreSpec
- union VixDiskLibSpec
- struct VixDiskLibConnectParams
- union VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds
- struct VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskL
- struct VixDiskLibConnectParams::VixDiskLibConnectParams::VixDiskLibCreds::VixDiskLibConnectParams::VixDiskL
- struct VixDiskLibInfo
- struct VixDiskLibBlock
- struct VixDiskLibBlockList

Defines

- #define VIXDISKLIB_SECTOR_SIZE 512
- #define VIXDISKLIB_HWVERSION_WORKSTATION_4 3
- #define VIXDISKLIB_HWVERSION_WORKSTATION_5 4
- #define VIXDISKLIB_HWVERSION_WORKSTATION_6 6
- #define VIXDISKLIB_HWVERSION_ESX30 VIXDISKLIB_HWVERSION_WORKSTATION_5
- #define VIXDISKLIB_HWVERSION_ESX4X 7
- #define VIXDISKLIB_HWVERSION_ESX50 8
- #define VIXDISKLIB_HWVERSION_ESX51 9
- #define VIXDISKLIB_HWVERSION_ESX55 10
- #define VIXDISKLIB_HWVERSION_ESX60 11
- #define VIXDISKLIB_HWVERSION_ESX65 13
- #define VIXDISKLIB_HWVERSION_CURRENT VIXDISKLIB_HWVERSION_ESX65
- #define VIXDISKLIB_MIN_CHUNK_SIZE (64 ∗ 2)
- #define VIXDISKLIB_MAX_CHUNK_SIZE (64 ∗ 2 ∗ 1024)
- #define VIXDISKLIB_MAX_CHUNK_NUMBER (512 ∗ 1024)

- #define VIXDISKLIB_FLAG_OPEN_UNBUFFERED (1 << 0)
- #define VIXDISKLIB_FLAG_OPEN_SINGLE_LINK (1 << 1)
- #define VIXDISKLIB_FLAG_OPEN_READ_ONLY (1 << 2)
- #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_ZLIB (1 << 4)
- #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_FASTLZ (1 << 5)
- #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_SKIPZ (1 << 6)
- #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_MASK (0x03f0)

Typedefs

- typedef uint64 VixDiskLibSectorType
- typedef struct VixDiskLibConnectParamsState VixDiskLibConnectParamsState
- typedef struct VixDiskLibHandleStruct VixDiskLibHandleStruct
- typedef VixDiskLibHandleStruct ∗ VixDiskLibHandle
- typedef struct VixDiskLibConnectParam ∗ VixDiskLibConnection
- typedef void(VixDiskLibGenericLogFunc)(const char ∗fmt, va_list args)
- typedef void(VixDiskLibGenericLogVFunc)(int routing, const char ∗fmt, va_list args)
- typedef Bool(∗ VixDiskLibProgressFunc)(void ∗progressData, int percentCompleted)
- typedef void(∗ VixDiskLibCompletionCB)(void ∗cbData, VixError result)

Enumerations

- enum VixDiskLibDiskType {

  VIXDISKLIB_DISK_MONOLITHIC_SPARSE = 1, VIXDISKLIB_DISK_MONOLITHIC_FLAT = 2, VIXDISKLIB_DISK_SPLIT_SPARSE = 3, VIXDISKLIB_DISK_SPLIT_FLAT = 4,

  VIXDISKLIB_DISK_VMFS_FLAT = 5, VIXDISKLIB_DISK_STREAM_OPTIMIZED = 6, VIXDISKLIB_DISK_VMFS_THIN = 7, VIXDISKLIB_DISK_VMFS_SPARSE = 8,

  VIXDISKLIB_DISK_UNKNOWN = 256 }
- enum VixDiskLibAdapterType { VIXDISKLIB_ADAPTER_IDE = 1, VIXDISKLIB_ADAPTER_SCSI_BUSLOGIC = 2, VIXDISKLIB_ADAPTER_SCSI_LSILOGIC = 3, VIXDISKLIB_ADAPTER_UNKNOWN = 256 }
- enum VixDiskLibCredType {

  VIXDISKLIB_CRED_UID = 1, VIXDISKLIB_CRED_SESSIONID = 2, VIXDISKLIB_CRED_TICKETID = 3, VIXDISKLIB_CRED_SSPI = 4,

  VIXDISKLIB_CRED_UNKNOWN = 256 }
- enum VixDiskLibSpecType { VIXDISKLIB_SPEC_VMX = 0, VIXDISKLIB_SPEC_VSTORAGE_OBJECT = 1, VIXDISKLIB_SPEC_DATASTORE = 128, VIXDISKLIB_SPEC_UNKNOWN = 256 }

Functions

- VixError VixDiskLib_InitEx (uint32 majorVersion, uint32 minorVersion, VixDiskLibGenericLogFunc ∗log, VixDiskLibGenericLogFunc ∗warn, VixDiskLibGenericLogFunc ∗panic, const char ∗libDir, const char ∗configFile)
- VixError VixDiskLib_Init (uint32 majorVersion, uint32 minorVersion, VixDiskLibGenericLogFunc ∗log, VixDiskLibGenericLogFunc ∗warn, VixDiskLibGenericLogFunc ∗panic, const char ∗libDir)
- void VixDiskLib_Exit (void)
- const char ∗ VixDiskLib_ListTransportModes (void)

- VixError VixDiskLib_Cleanup (const VixDiskLibConnectParams ∗connectParams, uint32 ∗numCleanedUp, uint32 ∗numRemaining)
- VixError VixDiskLib_Connect (const VixDiskLibConnectParams ∗connectParams, VixDiskLibConnection ∗connection)
- VixError VixDiskLib_PrepareForAccess (const VixDiskLibConnectParams ∗connectParams, const char ∗identity)
- VixError VixDiskLib_ConnectEx (const VixDiskLibConnectParams ∗connectParams, Bool readOnly, const char ∗snapshotRef, const char ∗transportModes, VixDiskLibConnection ∗connection)
- VixError VixDiskLib_Disconnect (VixDiskLibConnection connection)
- VixError VixDiskLib_EndAccess (const VixDiskLibConnectParams ∗connectParams, const char ∗identity)
- VixError VixDiskLib_Create (const VixDiskLibConnection connection, const char ∗path, const VixDiskLibCreateParams ∗createParams, VixDiskLibProgressFunc progressFunc, void ∗progressCallbackData)
- VixError VixDiskLib_CreateChild (VixDiskLibHandle diskHandle, const char ∗childPath, VixDiskLibDiskType diskType, VixDiskLibProgressFunc progressFunc, void ∗progressCallbackData)
- VixError VixDiskLib_Open (const VixDiskLibConnection connection, const char ∗path, uint32 flags, VixDiskLibHandle ∗diskHandle)
- VixError VixDiskLib_QueryAllocatedBlocks (VixDiskLibHandle diskHandle, VixDiskLibSectorType startSector, VixDiskLibSectorType numSectors, VixDiskLibSectorType chunkSize, VixDiskLibBlockList ∗∗blockList)
- VixError VixDiskLib_FreeBlockList (VixDiskLibBlockList ∗blockList)
- VixError VixDiskLib_GetInfo (VixDiskLibHandle diskHandle, VixDiskLibInfo ∗∗info)
- void VixDiskLib_FreeInfo (VixDiskLibInfo ∗info)
- const char ∗ VixDiskLib_GetTransportMode (VixDiskLibHandle diskHandle)
- VixError VixDiskLib_Close (VixDiskLibHandle diskHandle)
- VixError VixDiskLib_Read (VixDiskLibHandle diskHandle, VixDiskLibSectorType startSector, VixDiskLibSectorType numSectors, uint8 ∗readBuffer)
- VixError VixDiskLib_ReadAsync (VixDiskLibHandle diskHandle, VixDiskLibSectorType startSector, VixDiskLibSectorType numSectors, uint8 ∗readBuffer, VixDiskLibCompletionCB callback, void ∗cbData)
- VixError VixDiskLib_Write (VixDiskLibHandle diskHandle, VixDiskLibSectorType startSector, VixDiskLibSectorType numSectors, const uint8 ∗writeBuffer)
- VixError VixDiskLib_WriteAsync (VixDiskLibHandle diskHandle, VixDiskLibSectorType startSector, VixDiskLibSectorType numSectors, const uint8 ∗writeBuffer, VixDiskLibCompletionCB callback, void ∗cbData)
- VixError VixDiskLib_Flush (VixDiskLibHandle diskHandle)
- VixError VixDiskLib_Wait (VixDiskLibHandle diskHandle)
- VixError VixDiskLib_ReadMetadata (VixDiskLibHandle diskHandle, const char ∗key, char ∗buf, size_t bufLen, size_t ∗requiredLen)
- VixError VixDiskLib_WriteMetadata (VixDiskLibHandle diskHandle, const char ∗key, const char ∗val)
- VixError VixDiskLib_GetMetadataKeys (VixDiskLibHandle diskHandle, char ∗keys, size_t maxLen, size_t ∗requiredLen)
- VixError VixDiskLib_Unlink (VixDiskLibConnection connection, const char ∗path)
- VixError VixDiskLib_Grow (VixDiskLibConnection connection, const char ∗path, VixDiskLibSectorType capacity, Bool updateGeometry, VixDiskLibProgressFunc progressFunc, void ∗progressCallbackData)
- VixError VixDiskLib_Shrink (VixDiskLibHandle diskHandle, VixDiskLibProgressFunc progressFunc, void ∗progressCallbackData)

- VixError VixDiskLib_Defragment (VixDiskLibHandle diskHandle, VixDiskLibProgressFunc progressFunc, void ∗progressCallbackData)

- VixError VixDiskLib_Rename (const char ∗srcFileName, const char ∗dstFileName)

- VixError VixDiskLib_Clone (const VixDiskLibConnection dstConnection, const char ∗dstPath, const VixDiskLibConnection srcConnection, const char ∗srcPath, const VixDiskLibCreateParams ∗vixCreateParams, VixDiskLibProgressFunc progressFunc, void ∗progressCallbackData, Bool overWrite)

- char ∗ VixDiskLib_GetErrorText (VixError err, const char ∗locale)

- void VixDiskLib_FreeErrorText (char ∗errMsg)

- VixError VixDiskLib_IsAttachPossible (VixDiskLibHandle parent, VixDiskLibHandle child)

- VixError VixDiskLib_Attach (VixDiskLibHandle parent, VixDiskLibHandle child)

- VixError VixDiskLib_SpaceNeededForClone (VixDiskLibHandle diskHandle, VixDiskLibDiskType cloneDiskType, uint64 ∗spaceNeeded)

- VixError VixDiskLib_CheckRepair (const VixDiskLibConnection connection, const char ∗filename, Bool repair)

- VixError VixDiskLib_GetConnectParams (const VixDiskLibConnection connection, VixDiskLibConnectParams ∗∗connectParams)

- void VixDiskLib_FreeConnectParams (VixDiskLibConnectParams ∗connectParams)

- VixDiskLibConnectParams ∗ VixDiskLib_AllocateConnectParams ()

4.1.1 Define Documentation

4.1.1.1 #define VIXDISKLIB_SECTOR_SIZE 512

4.1.1.2 #define VIXDISKLIB_HWVERSION_WORKSTATION_4 3

4.1.1.3 #define VIXDISKLIB_HWVERSION_WORKSTATION_5 4

4.1.1.4 #define VIXDISKLIB_HWVERSION_WORKSTATION_6 6

4.1.1.5 #define VIXDISKLIB_HWVERSION_ESX30 VIXDISKLIB_HWVERSION_-WORKSTATION_5

4.1.1.6 #define VIXDISKLIB_HWVERSION_ESX4X 7

4.1.1.7 #define VIXDISKLIB_HWVERSION_ESX50 8

4.1.1.8 #define VIXDISKLIB_HWVERSION_ESX51 9

4.1.1.9 #define VIXDISKLIB_HWVERSION_ESX55 10

4.1.1.10 #define VIXDISKLIB_HWVERSION_ESX60 11

4.1.1.11 #define VIXDISKLIB_HWVERSION_ESX65 13

4.1.1.12 #define VIXDISKLIB_HWVERSION_CURRENT VIXDISKLIB_HWVERSION_ESX65

4.1.1.13 #define VIXDISKLIB_MIN_CHUNK_SIZE (64 ∗ 2)

4.1.1.14 #define VIXDISKLIB_MAX_CHUNK_SIZE (64 ∗ 2 ∗ 1024)

4.1.1.15 #define VIXDISKLIB_MAX_CHUNK_NUMBER (512 ∗ 1024)

4.1.1.16 #define VIXDISKLIB_FLAG_OPEN_UNBUFFERED (1 << 0)

4.1.1.17 #define VIXDISKLIB_FLAG_OPEN_SINGLE_LINK (1 << 1)

4.1.1.18 #define VIXDISKLIB_FLAG_OPEN_READ_ONLY (1 << 2)

4.1.1.19 #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_ZLIB (1 << 4)

4.1.1.20 #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_FASTLZ (1 << 5)

4.1.1.21 #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_SKIPZ (1 << 6)

4.1.1.22 #define VIXDISKLIB_FLAG_OPEN_COMPRESSION_MASK (0x03f0)

4.1.2 Typedef Documentation

4.1.2.1 typedef uint64 VixDiskLibSectorType

4.1.2.2 typedef struct VixDiskLibConnectParamsState VixDiskLibConnectParamsState

4.1.2.3 typedef struct VixDiskLibHandleStruct VixDiskLibHandleStruct

4.1.2.4 typedef VixDiskLibHandleStruct∗ VixDiskLibHandle

4.1.2.5 typedef struct VixDiskLibConnectParam∗ VixDiskLibConnection

4.1.2.6 typedef void(VixDiskLibGenericLogFunc)(const char ∗fmt, va_list args)

VIXDISKLIB_DISK_MONOLITHIC_FLAT
VIXDISKLIB_DISK_SPLIT_SPARSE
VIXDISKLIB_DISK_SPLIT_FLAT
VIXDISKLIB_DISK_VMFS_FLAT
VIXDISKLIB_DISK_STREAM_OPTIMIZED
VIXDISKLIB_DISK_VMFS_THIN
VIXDISKLIB_DISK_VMFS_SPARSE
VIXDISKLIB_DISK_UNKNOWN

4.1.3.2 enum VixDiskLibAdapterType

Enumerator:

VIXDISKLIB_ADAPTER_IDE
VIXDISKLIB_ADAPTER_SCSI_BUSLOGIC
VIXDISKLIB_ADAPTER_SCSI_LSILOGIC
VIXDISKLIB_ADAPTER_UNKNOWN

4.1.3.3 enum VixDiskLibCredType

Enumerator:

VIXDISKLIB_CRED_UID
VIXDISKLIB_CRED_SESSIONID
VIXDISKLIB_CRED_TICKETID
VIXDISKLIB_CRED_SSPI
VIXDISKLIB_CRED_UNKNOWN

4.1.3.4 enum VixDiskLibSpecType

the type of spec

Enumerator:

VIXDISKLIB_SPEC_VMX
VIXDISKLIB_SPEC_VSTORAGE_OBJECT
VIXDISKLIB_SPEC_DATASTORE
VIXDISKLIB_SPEC_UNKNOWN

4.1.4 Function Documentation

4.1.4.1 VixError VixDiskLib_InitEx (uint32 *majorVersion*, uint32 *minorVersion*, VixDiskLibGenericLogFunc ∗ *log*, VixDiskLibGenericLogFunc ∗ *warn*, VixDiskLibGenericLogFunc ∗ *panic*, const char ∗ *libDir*, const char ∗ *configFile*)

Initializes VixDiskLib.

Parameters:

majorVersion [in] Required major version number for client.

minorVersion [in] Required minor version number for client.

log [in] Callback for Log entries.

warn [in] Callback for warnings.

panic [in] Callback for panic.

libDir [in] Directory location where dependent libs are located.

configFile [in] Configuration file path in local encoding. configuration files are of the format name = "value" each name/value pair on a separate line. For a detailed description of allowed values, refer to the VixDiskLib documentation.

Returns:

VIX_OK on success, suitable VIX error code otherwise.

4.1.4.2 VixError VixDiskLib_Init (uint32 *majorVersion*, uint32 *minorVersion*, VixDiskLibGenericLogFunc ∗ *log*, VixDiskLibGenericLogFunc ∗ *warn*, VixDiskLibGenericLogFunc ∗ *panic*, const char ∗ *libDir*)

Initializes VixDiskLib - deprecated, please use VixDiskLib_InitEx.

Parameters:

majorVersion [in] Required major version number for client.

minorVersion [in] Required minor version number for client.

log [in] Callback for Log entries.

warn [in] Callback for warnings.

panic [in] Callback for panic.

libDir [in] Directory location where dependent libs are located.

Returns:

VIX_OK on success, suitable VIX error code otherwise.

4.1.4.3 void VixDiskLib_Exit (void)

Cleans up VixDiskLib.

4.1.4.4 const char∗ VixDiskLib_ListTransportModes (void)

Get a list of transport modes known to VixDiskLib. This list is also the default used if VixDiskLib_ConnectEx is called with transportModes set to NULL.

The string is a list of transport modes separated by colons. For example: "file:san:hotadd:nbd". See VixDiskLib_ConnectEx for more details.

Returns:

Returns a string that is a list of plugins. The caller must not free the string.

4.1.4.5 VixError VixDiskLib_Cleanup (const VixDiskLibConnectParams ∗ *connectParams*, uint32 ∗ *numCleanedUp*, uint32 ∗ *numRemaining*)

Perform a cleanup after an unclean shutdown of an application using VixDiskLib.

When using VixDiskLib_ConnectEx, some state might have not been cleaned up if the resulting connection was not shut down cleanly. Use VixDiskLib_Cleanup to remove this extra state.

Parameters:

connectParams [in] Hostname and login credentials to connect to a host managing virtual machines that were accessed and need cleanup. While VixDiskLib_Cleanup can be invoked for local connections as well, it is a no-op in that case. Also, the vmxSpec property of connectParams should be set to NULL.

numCleanedUp [out] Number of virtual machines that were successfully cleaned up. – Can be NULL.

numRemaining [out] Number of virutal machines that still require cleaning up. – Can be NULL.

Returns:

VIX_OK if all virtual machines were successfully cleaned up or if no virtual machines required cleanup. VIX error code otherwise and numRemaning can be used to check for the number of virtual machines requiring cleanup.

4.1.4.6 VixError VixDiskLib_Connect (const VixDiskLibConnectParams ∗ *connectParams*, VixDiskLibConnection ∗ *connection*)

Connects to a local / remote server.

Parameters:

connectParams [in] NULL if manipulating local disks. For remote case this includes esx hostName and user credentials.

connection [out] Returned handle to a connection.

Returns:

VIX_OK if success suitable VIX error code otherwise.

4.1.4.7 VixError VixDiskLib_PrepareForAccess (const VixDiskLibConnectParams ∗ *connectParams*, const char ∗ *identity*)

This function is used to notify the host of the virtual machine that the disks of the virtual machine will be opened. The host disables operations on the virtual machine that may be adversely affected if they are performed while the disks are open by a third party application.

Parameters:

connectParams [in] This is always used on remote connections.

identity [in] An arbitrary string containing the identity of the application.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.8 VixError VixDiskLib_ConnectEx (const VixDiskLibConnectParams $*$ *connectParams*, Bool *readOnly*, const char $*$ *snapshotRef*, const char $*$ *transportModes*, VixDiskLibConnection $*$ *connection*)

Create a transport context to access disks belonging to a particular snapshot of a particular virtual machine. Using this transport context will enable callers to open virtual disks using the most efficient data acces protocol available for managed virtual machines, hence getting better I/O performance.

If this call is used instead of VixDiskLib_Connect, the additional information passed in will be used in order to optimize the I/O access path, to maximize I/O throughput.

Note: For local virtual machines/disks, this call is equivalent to VixDiskLib_Connect.

Parameters:

connectParams [in] NULL if maniuplating local disks. For remote case this includes esx hostName and user credentials.

readOnly [in] Should be set to TRUE if no write access is needed for the disks to be accessed through this connection. In some cases, a more efficient I/O path can be used for read-only access.

snapshotRef [in] A managed object reference to the specific snapshot of the virtual machine whose disks will be accessed with this connection. Specifying this property is only meaningful if the vmxSpec property in connectParams is set as well.

transportModes [in] An optional list of transport modes that can be used for this connection, separated by colons. If NULL is specified, VixDiskLib's default setting of "file:san:hotadd:nbd" is used. If a disk is opened through this connection, VixDiskLib will start with the first entry of the list and attempt to use this transport mode to gain access to the virtual disk. If this does not work, the next item in the list will be used until either the disk was successfully opened or the end of the list is reached.

connection [out] Returned handle to a connection.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.9 VixError VixDiskLib_Disconnect (VixDiskLibConnection *connection*)

Breaks an existing connection.

Parameters:

connection [in] Valid handle to a (local/remote) connection.

Returns:

VIX_OK if success suitable VIX error code otherwise.

4.1.4.10 VixError VixDiskLib_EndAccess (const VixDiskLibConnectParams $*$ *connectParams*, const char $*$ *identity*)

This function is used to notify the host of a virtual machine that the virtual machine disks are closed and that the operations which rely on the virtual machine disks to be closed can now be allowed.

Parameters:

connectParams [in] Always used for a remote connection. Must be the same parameters as used in the corresponding PrepareForAccess call.

identity [in] An arbitrary string containing the identity of the application.

Returns:

VIX_OK of success, suitable VIX error code otherwise.

4.1.4.11 VixError VixDiskLib_Create (const VixDiskLibConnection *connection*, const char ∗ *path*, const VixDiskLibCreateParams ∗ *createParams*, VixDiskLibProgressFunc *progressFunc*, void ∗ *progressCallbackData*)

Creates a local disk. Remote disk creation is not supported.

Parameters:

connection [in] A valid connection.

path [in] VMDK file name given as absolute path e.g. "c:\\My Virtual Machines\\MailServer\SystemDisk.vmdk".

createParams [in] Specification for the new disk (type, capacity ...).

progressFunc [in] Callback to report progress.

progressCallbackData [in] Callback data pointer.

Returns:

VIX_OK if success suitable VIX error code otherwise.

4.1.4.12 VixError VixDiskLib_CreateChild (VixDiskLibHandle *diskHandle*, const char ∗ *childPath*, VixDiskLibDiskType *diskType*, VixDiskLibProgressFunc *progressFunc*, void ∗ *progressCallbackData*)

Creates a redo log from a parent disk.

Parameters:

diskHandle [in] Handle to an open virtual disk.

childPath [in] Redo log file name given as absolute path e.g. "c:\\My Virtual Machines\\MailServer\SystemDisk_s0001.vmdk".

diskType [in] Either VIXDISKLIB_DISK_MONOLITHIC_SPARSE or VIXDISKLIB_DISK_-SPLIT_SPARSE.

progressFunc [in] Callback to report progress.

progressCallbackData [in] Callback data pointer.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.13 VixError VixDiskLib_Open (const VixDiskLibConnection *connection*, const char $*$ *path*, uint32 *flags*, VixDiskLibHandle $*$ *diskHandle*)

Opens a local or remote virtual disk.

Parameters:

connection [in] A valid connection.

path [in] VMDK file name given as absolute path e.g. "[storage1] MailServer/SystemDisk.vmdk"

flags [in, optional] Bitwise or'ed combination of VIXDISKLIB_FLAG_OPEN_UNBUFFERED VIXDISKLIB_FLAG_OPEN_SINGLE_LINK VIXDISKLIB_FLAG_OPEN_READ_ONLY.

diskHandle [out] Handle to opened disk, NULL if disk was not opened.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.14 VixError VixDiskLib_QueryAllocatedBlocks (VixDiskLibHandle *diskHandle*, VixDiskLibSectorType *startSector*, VixDiskLibSectorType *numSectors*, VixDiskLibSectorType *chunkSize*, VixDiskLibBlockList $**$ *blockList*)

Get the blocks allocated.

Parameters:

diskHandle [in] Handle to an open virtual disk.

startSector [in] Absolute offset.

numSectors [in] Number of sectors to query.

chunkSize [in] Minimum number of sectors covered by a chunk containing data.

blockList [out] Buffer contains a VixDiskLibBlockList.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.15 VixError VixDiskLib_FreeBlockList (VixDiskLibBlockList $*$ *blockList*)

Frees memory allocated in VixDiskLib_QueryAllocatedBlocks.

Parameters:

blockList [in] block list to be freed.

4.1.4.16 VixError VixDiskLib_GetInfo (VixDiskLibHandle *diskHandle*, VixDiskLibInfo $**$ *info*)

Retrieves information about a disk.

Parameters:

diskHandle [in] Handle to an open virtual disk.

info [out] Disk information filled up.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.17 void VixDiskLib_FreeInfo (VixDiskLibInfo ∗ *info*)

Frees memory allocated in VixDiskLib_GetInfo.

Parameters:

info [in] Disk information to be freed.

4.1.4.18 const char∗ VixDiskLib_GetTransportMode (VixDiskLibHandle *diskHandle*)

Returns a pointer to a static string identifying the transport mode that is used to access the virtual disk's data.

If a disk was opened through a connection obtained by VixDiskLib_Connect, the return value will be "file" for a local disk and "nbd" or "nbdssl" for a managed disk.

The pointer to this string is static and must not be deallocated by the caller.

Parameters:

diskHandle [in] Handle to an open virtual disk.

Returns:

Returns a pointer to a static string identifying the transport mode used to access the disk's data.

4.1.4.19 VixError VixDiskLib_Close (VixDiskLibHandle *diskHandle*)

Closes the disk.

Parameters:

diskHandle [in] Handle to an open virtual disk.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.20 VixError VixDiskLib_Read (VixDiskLibHandle *diskHandle*, VixDiskLibSectorType *startSector*, VixDiskLibSectorType *numSectors*, uint8 ∗ *readBuffer*)

Reads a sector range.

Parameters:

diskHandle [in] Handle to an open virtual disk.

startSector [in] Absolute offset.

numSectors [in] Number of sectors to read.

readBuffer [out] Buffer to read into.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.21 VixError VixDiskLib_ReadAsync (VixDiskLibHandle *diskHandle*, VixDiskLibSectorType *startSector*, VixDiskLibSectorType *numSectors*, uint8 ∗ *readBuffer*, VixDiskLibCompletionCB *callback*, void ∗ *cbData*)

Reads a sector range asynchronously.

Parameters:

diskHandle [in] Handle to an open virtual disk.

startSector [in] Absolute offset.

numSectors [in] Number of sectors to read.

readBuffer [out] Buffer to read into.

callback [in] Callback when data has been read

cbData [in] User context data supplied during the callback

Returns:

VIX_ASYNC if success, suitable VIX error code otherwise.

4.1.4.22 VixError VixDiskLib_Write (VixDiskLibHandle *diskHandle*, VixDiskLibSectorType *startSector*, VixDiskLibSectorType *numSectors*, const uint8 ∗ *writeBuffer*)

Writes a sector range.

Parameters:

diskHandle [in] Handle to an open virtual disk.

startSector [in] Absolute offset.

numSectors [in] Number of sectors to write.

writeBuffer [in] Buffer to write.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.23 VixError VixDiskLib_WriteAsync (VixDiskLibHandle *diskHandle*, VixDiskLibSectorType *startSector*, VixDiskLibSectorType *numSectors*, const uint8 ∗ *writeBuffer*, VixDiskLibCompletionCB *callback*, void ∗ *cbData*)

Writes a sector range asynchronously.

Parameters:

diskHandle [in] Handle to an open virtual disk.

startSector [in] Absolute offset.

numSectors [in] Number of sectors to write.

writeBuffer [in] Buffer to write.

callback [in] Callback when data has been written/completed

cbData [in] User context data supplied during the callback

Returns:

VIX_ASYNC if success, suitable VIX error code otherwise.

4.1.4.24 VixError VixDiskLib_Flush (VixDiskLibHandle *diskHandle*)

Flush the async write data to disk

Parameters:

diskHandle [in] Handle to an open virtual disk.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.25 VixError VixDiskLib_Wait (VixDiskLibHandle *diskHandle*)

Waits for all async operations to complete

Parameters:

diskHandle [in] Handle to an open virtual disk.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.26 VixError VixDiskLib_ReadMetadata (VixDiskLibHandle *diskHandle*, const char ∗ *key*, char ∗ *buf*, size_t *bufLen*, size_t ∗ *requiredLen*)

Retrieves the value of a metadata entry corresponding to the supplied key.

Parameters:

diskHandle [in] Handle to an open virtual disk.

key [in] Key name.

buf [out, optional] Placeholder for key's value in the metadata store, can be NULL.

bufLen [in] Size of the buffer.

requiredLen [out, optional] Size of buffer required for the value (including end of string character)

Returns:

VIX_OK if success, VIX_E_DISK_BUFFER_TOO_SMALL if too small a buffer and other errors as applicable.

4.1.4.27 VixError VixDiskLib_WriteMetadata (VixDiskLibHandle *diskHandle*, const char ∗ *key*, const char ∗ *val*)

Creates or modifies a metadata table entry.

Parameters:

diskHandle [in] Handle to an open virtual disk.

key [in] Key name.

val [in] Key's value.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.28 VixError VixDiskLib_GetMetadataKeys (VixDiskLibHandle *diskHandle*, char ∗ *keys*, size_t *maxLen*, size_t ∗ *requiredLen*)

Retrieves the list of keys in the metadata table. Key names are returned as list of null-terminated strings, followed by an additional NULL character.

Parameters:

diskHandle [in] Handle to an open virtual disk.

keys [out, optional] Keynames buffer, can be NULL.

maxLen [in] Size of the keynames buffer.

requiredLen [out, optional] Space required for the keys including the double end-of-string characters.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.29 VixError VixDiskLib_Unlink (VixDiskLibConnection *connection*, const char ∗ *path*)

Deletes all extents of the specified disk link. If the path refers to a parent disk, the child (redo log) will be orphaned. Unlinking the child does not affect the parent.

Parameters:

connection [in] A valid connection.

path [in] Path to the disk to be deleted.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.30 VixError VixDiskLib_Grow (VixDiskLibConnection *connection*, const char ∗ *path*, VixDiskLibSectorType *capacity*, Bool *updateGeometry*, VixDiskLibProgressFunc *progressFunc*, void ∗ *progressCallbackData*)

Grows an existing disk, only local disks are grown.

Precondition:

The specified disk is not open.

Parameters:

connection [in] A valid connection.

path [in] Path to the disk to be grown.

capacity [in] Target size for the disk.

updateGeometry [in] Should vixDiskLib update the geometry?

progressFunc [in] Callback to report progress (called on the same thread).

progressCallbackData [in] Opaque pointer passed along with the percent complete.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.31 VixError VixDiskLib_Shrink (VixDiskLibHandle *diskHandle*, VixDiskLibProgressFunc *progressFunc*, void ∗ *progressCallbackData*)

Shrinks an existing disk, only local disks are shrunk.

Parameters:

diskHandle [in] Handle to an open virtual disk.

progressFunc [in] Callback to report progress (called on the same thread).

progressCallbackData [in] Opaque pointer passed along with the percent complete.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.32 VixError VixDiskLib_Defragment (VixDiskLibHandle *diskHandle*, VixDiskLibProgressFunc *progressFunc*, void ∗ *progressCallbackData*)

Defragments an existing disk.

Parameters:

diskHandle [in] Handle to an open virtual disk.

progressFunc [in] Callback to report progress (called on the same thread).

progressCallbackData [in] Opaque pointer passed along with the percent complete.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.33 VixError VixDiskLib_Rename (const char ∗ *srcFileName*, const char ∗ *dstFileName*)

Renames a virtual disk.

Parameters:

srcFileName [in] Virtual disk file to rename.

dstFileName [in] New name for the virtual disk.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.34 VixError VixDiskLib_Clone (const VixDiskLibConnection *dstConnection*, const char ∗ *dstPath*, const VixDiskLibConnection *srcConnection*, const char ∗ *srcPath*, const VixDiskLibCreateParams ∗ *vixCreateParams*, VixDiskLibProgressFunc *progressFunc*, void ∗ *progressCallbackData*, Bool *overWrite*)

Copies a disk with proper conversion.

Parameters:

dstConnection [in] A valid connection to access the destination disk.

dstPath [in] Absolute path for the (new) destination disk.

srcConnection [in] A valid connection to access the source disk.

srcPath [in] Absolute path for the source disk.

vixCreateParams [in] creationParameters (disktype, hardware type...). If the destination is remote, createParams is currently ignored and disk with default size and adapter type is created.

progressFunc [in] Callback to report progress (called on the same thread).

progressCallbackData [in] Opaque pointer passed along with the percent complete.

overWrite [in] TRUE if Clone should overwrite an existing file.

Returns:

VIX_OK if success, suitable VIX error code otherwise (network errors like file already exists handshake failure, ... are all combined into a generic connect message).

4.1.4.35 char∗ VixDiskLib_GetErrorText (VixError *err*, const char ∗ *locale*)

Returns the textual description of an error.

Parameters:

err [in] A VIX error code.

locale [in] Language locale - not currently supported and must be NULL.

Returns:

The error message string. This should only be deallocated by VixDiskLib_FreeErrorText. Returns NULL if there is an error in retrieving text.

4.1.4.36 void VixDiskLib_FreeErrorText (char ∗ *errMsg*)

Free the error message returned by VixDiskLib_GetErrorText.

Parameters:

errMsg [in] Message string returned by VixDiskLib_GetErrorText. It is OK to call this function with NULL.

Returns:

None.

4.1.4.37 VixError VixDiskLib_IsAttachPossible (VixDiskLibHandle *parent*, VixDiskLibHandle *child*)

Checks if the child disk chain can be attached to the parent disk cahin.

Parameters:

parent [in] Handle to the disk to be attached.

child [in] Handle to the disk to attach.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.38 VixError VixDiskLib_Attach (VixDiskLibHandle *parent*, VixDiskLibHandle *child*)

Attaches the child disk chain to the parent disk chain. Parent handle is invalid after attaching and child represents the combined disk chain.

Parameters:

parent [in] Handle to the disk to be attached.

child [in] Handle to the disk to attach.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.39 VixError VixDiskLib_SpaceNeededForClone (VixDiskLibHandle *diskHandle*, VixDiskLibDiskType *cloneDiskType*, uint64 ∗ *spaceNeeded*)

Compute the space (in bytes) required to copy a disk chain.

Parameters:

diskHandle [in] Handle to the disk to be copied.

cloneDiskType [in] Type of the (to be) newly created disk. If cloneDiskType is VIXDISKLIB_DISK_UNKNOWN, the source disk type is assumed.

spaceNeeded [out] Place holder for space needed in bytes.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.40 VixError VixDiskLib_CheckRepair (const VixDiskLibConnection *connection*, const char ∗ *filename*, Bool *repair*)

Check a sparse disk for internal consistency.

Parameters:

connection [in] A VixDiskLib connection.

filename [in] Path to disk to be checked.

repair [in] TRUE if repair should be attempted, false otherwise.

Returns:

VIX_OK if success, suitable VIX error code otherwise. Note this refers to the success of the call, not the consistency of the disk being checked.

4.1.4.41 VixError VixDiskLib_GetConnectParams (const VixDiskLibConnection *connection*, VixDiskLibConnectParams ** *connectParams*)

Return the details for the connection.

Parameters:

connection [in] A VixDiskLib connection.

connectParams [out] Details of the connection.

Returns:

VIX_OK if success, suitable VIX error code otherwise.

4.1.4.42 void VixDiskLib_FreeConnectParams (VixDiskLibConnectParams * *connectParams*)

Free the connection details structure allocated during VixDiskLib_GetConnectParams or VixDiskLib_-AllocateConnectParams.

Parameters:

connectParams [out] Connection details to be free'ed.

Returns:

None.

4.1.4.43 VixDiskLibConnectParams* VixDiskLib_AllocateConnectParams ()

Allocate the connection details structure

Returns:

A pointer to the instance of connection details; NULL if error.

4.2 distribute_vixDiskLibNasPlugin.h File Reference

Data Structures

- struct VixDiskLibNasPluginDataStoreParams
- struct VixDiskLibNasPluginSessionParams
- struct VixDiskLibNasPluginProgressRecord
- struct VixDiskLibNasPluginResultCommon
- struct VixDiskLibNasPluginStatXResult
- struct VixDiskLibNasPluginCommonParams
- struct VixDiskLibNasPluginCloneFileParams
- struct VixDiskLibNasPluginResvSpaceParams
- struct VixDiskLibNasPluginStatXParams
- struct VixDiskLibNasPlugin

Defines

- #define VIXDISKLIB_NASPLUGIN_MAJOR_VERSION 1
- #define VIXDISKLIB_NASPLUGIN_MINOR_VERSION 2
- #define VIXDISKLIB_NASPLUGIN_FSTYPE_NFS "NFS"
- #define VIXDISKLIB_NASPLUGIN_FSTYPE_NFS41 "NFS41"
- #define VIXDISKLIB_NASPLUGIN_FSTYPE_VMFS "VMFS"
- #define VIXDISKLIB_NASPLUGIN_FSTYPE_VMFSL "VMFS-L"
- #define VIXDISKLIB_NASPLUGIN_INVALID_SESSION_ID (0LL)

Typedefs

- typedef uint64 VixDiskLibNasPluginSessionID
- typedef Bool(VixDiskLibNasPluginPeriodicCallback)(struct VixDiskLibNasPluginProgressRecord ∗pRec)
- typedef VixError(VixDiskLibNasPluginStartSession)(const VixDiskLibNasPluginDataStoreParams ∗nasConfigData, const VixDiskLibNasPluginSessionParams ∗sessionParams, VixDiskLibNasPluginSessionID ∗sessionID)
- typedef VixError(VixDiskLibNasPluginEndSession)(VixDiskLibNasPluginSessionID sessionID)
- typedef VixError(VixDiskLibNasPluginSupportStatus)(VixDiskLibNasPluginSessionID sessionID, const VixDiskLibNasPluginPrimitiveID primitiveID)
- typedef void(VixDiskLibNasPluginExecutePrimitive)(VixDiskLibNasPluginSessionID sessionID, const VixDiskLibNasPluginCommonParams ∗execParams)

Enumerations

- enum VixDiskLibNasPluginPrimitiveID { VIXDISKLIB_NASPLUGIN_PRIM_INVALID = ((’N’ $<<$ 24) | (’A’ $<<$ 16) | (’S’ $<<$ 8) | ’0’), VIXDISKLIB_NASPLUGIN_PRIM_CLONEFILE, VIXDISKLIB_NASPLUGIN_PRIM_RESVSPACE, VIXDISKLIB_NASPLUGIN_PRIM_STATX }
- enum VixDiskLibNasPluginAllocType { VIXDISKLIB_NASPLUGIN_FILE_ALLOC_EZT, VIXDISKLIB_NASPLUGIN_FILE_ALLOC_LZT, VIXDISKLIB_NASPLUGIN_FILE_ALLOC_THIN, VIXDISKLIB_NASPLUGIN_FILE_ALLOC_UNKNOWN = -1 }

- enum VixDiskLibNasPluginCloneFileFlags {

  VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_GUARDED = (1 << 0), VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_LAZY = (1 << 1), VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_SRCDATASTORE_VALID = (1 << 2), VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_DRYRUN = (1 << 3),

  VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_SKIPZEROES = (1 << 4) }

4.2.1 Define Documentation

4.2.1.1 #define VIXDISKLIB_NASPLUGIN_MAJOR_VERSION 1

4.2.1.2 #define VIXDISKLIB_NASPLUGIN_MINOR_VERSION 2

4.2.1.3 #define VIXDISKLIB_NASPLUGIN_FSTYPE_NFS "NFS"

4.2.1.4 #define VIXDISKLIB_NASPLUGIN_FSTYPE_NFS41 "NFS41"

4.2.1.5 #define VIXDISKLIB_NASPLUGIN_FSTYPE_VMFS "VMFS"

4.2.1.6 #define VIXDISKLIB_NASPLUGIN_FSTYPE_VMFSL "VMFS-L"

4.2.1.7 #define VIXDISKLIB_NASPLUGIN_INVALID_SESSION_ID (0LL)

4.2.2 Typedef Documentation

4.2.2.1 typedef uint64 VixDiskLibNasPluginSessionID

4.2.2.2 typedef Bool(VixDiskLibNasPluginPeriodicCallback)(struct VixDiskLibNasPluginProgressRecord ∗pRec)

4.2.2.3 typedef VixError(VixDiskLibNasPluginStartSession)(const VixDiskLibNasPluginDataStoreParams ∗nasConfigData, const VixDiskLibNasPluginSessionParams ∗sessionParams, VixDiskLibNasPluginSessionID ∗sessionID)

Prototype for VixDiskLibNasPlugin session initialization.

The NAS plugin needs configuration information to establish connectivity to the NAS device. VI already stores this information as part of NFS datastore configuration for VMFS. The NAS plugin is provided NAS configuration information as part of an init session request. The configuration data contains the NAS device IP address, NAS device mount point, corresponding datastore name on the ESX host, and the ESX filesystem type and version. This should enable a NAS plugin writer to connect to, say, a RPC program running on the NAS device and exchange other vendor-specific setup information and configuration data. The plugin should return a session identifier for the virtual disk library to use as a context for a single or a set of VM operations. The client can also negotiate session properties like timeouts, priority etc. during session initiation. The session initialization may fail if the plugin does not support the given NAS volume.

Note that this entry point may be called for a datastore which the plugin does not support. For example, it may be a called for a datastore on a NAS server from a different vendor, or datastore on a NAS server whose firmware does not support NAS offloads. It is the plugin's responsibility to verify it can communicate with the NAS server and that the NAS server supports the offload primitives and that the remote mount point exists on the NAS server.

Parameters:

nasConfigData IN The configuration data contains the NAS device IP address, NAS device mount point, corresponding datastore name on the ESX host, and the ESX filesystem type and version.

flags IN sessionParams Session properties like timeouts, priority

sessionID OUT Structure returned in the case of a successful session; it is an identifier for the virtual disk library to use as a context for a single or a set of VM operations.

Returns:

VIX_OK if the session was successfully established, a VIX_* error otherwise.

4.2.2.4 typedef VixError(VixDiskLibNasPluginEndSession)(VixDiskLibNasPluginSessionID sessionID)

Prototype for VixDiskLibNasPlugin session teardown.

The caller may issue multiple sets of VixDiskLibNasPluginExecutePrimitive calls in a given session. A session may end at the request of the caller or after the plugin hits a timeout.

Parameters:

VixDiskLibNasPluginSessionID sessionID IN Structure that was returned by the corresponding VixDiskLibNasPluginStartSession call.

Returns:

VIX_OK if the session was successfully terminated, a VIX_* error otherwise.

4.2.2.5 typedef VixError(VixDiskLibNasPluginSupportStatus)(VixDiskLibNasPluginSessionID sessionID, const VixDiskLibNasPluginPrimitiveID primitiveID)

Prototype for VixDiskLibNasPlugin entrypoint to check if a primitive is supported.

Once a session is established, the caller can ask for the support status of a vStorage primitive in the context of an established session ID. For example, the caller may ask for support status of the VIXDISKLIB_-NASPLUGIN_PRIM_STATX primitive. The plugin shall determine if the offload primitive is supported on the datastore. This may require communication with the NAS device, and the method is vendor specific.

Parameters:

sessionID IN Handle identifying a live session

primitiveID IN Primitive ID to get the support status for.

Returns:

VIX_OK if the session support was successfully determined, a VIX_* error otherwise.

4.2.2.6 typedef void(VixDiskLibNasPluginExecutePrimitive)(VixDiskLibNasPluginSessionID sessionID, const VixDiskLibNasPluginCommonParams *execParams)

Prototype for VixDiskLibNasPlugin primitive execution.

Once a session is established, the caller can ask for a vStorage primitive to be done on a set of files in the context of an established session ID. For example, the caller may ask for a file to be quick-cloned (snapshotted) to another file on the same datastore. The plugin shall issue the request to the NAS device in a vendor specific format.

One example of a primitive that would be very useful on NFS is the ability to reserve physical space to back the logical size of a file. The preferred disk format to run enterprise workloads in ESX virtual machines is zeroedthick; VMFS-3 reserves space for zeroedthick virtual disk files at file creation time, thus removing the possibility of the virtual machine crashing because of lack of physical space at the time the application writes to the virtual disk. A newly created virtual disk file on a NFS server could be set to an arbitrary length (many tens or hundreds of GB in this case) using POSIX lseek and write, but the NFS server will not allocate backing storage to the file. This violates the space guarantee requirement of zeroedthick disk format. On NAS devices that understand the vStorage API for file devices, one can issue an VixDiskLibNasPluginExecutePrimitive with primitiveID set to VIXDISKLIB_NASPLUGIN_PRIM_-RESVSPACE to instruct the NAS device to use vendor-specific mechanisms to reserve space for a newly created or existing virtual disk of non-zero logical size.

This entrypoint can block until the client request is successfully completed on the NAS device. However, the plugin must periodically invoke the provided update callback. Specifically, the plugin must invoke execParams->progressRecord->callback for every execParams->progressRecord->updateBytes bytes that it transfers, each time setting the execParams->progressRecord->progressBytes appropriately. If the callback returns FALSE, it is an indication that the user has requested to abort the operation; as such, the plugin must abort the operation (including the cleanup of any internal state), set the execParams->result data appropriately, and return. If communication with the NAS device is broken or timed out or no progress is reported by the NAS device within the session timeout period, the NAS plugin must abort the operation and return.

Parameters:

sessionID IN Handle identifying a live session

execParams IN Parameters to the operation.

4.2.3 Enumeration Type Documentation

4.2.3.1 enum VixDiskLibNasPluginPrimitiveID

Enumerator:

VIXDISKLIB_NASPLUGIN_PRIM_INVALID

VIXDISKLIB_NASPLUGIN_PRIM_CLONEFILE

VIXDISKLIB_NASPLUGIN_PRIM_RESVSPACE

VIXDISKLIB_NASPLUGIN_PRIM_STATX

4.2.3.2 enum VixDiskLibNasPluginAllocType

Enumerator:

VIXDISKLIB_NASPLUGIN_FILE_ALLOC_EZT

VIXDISKLIB_NASPLUGIN_FILE_ALLOC_LZT

VIXDISKLIB_NASPLUGIN_FILE_ALLOC_THIN

VIXDISKLIB_NASPLUGIN_FILE_ALLOC_UNKNOWN

4.2.3.3 enum VixDiskLibNasPluginCloneFileFlags

Enumerator:

VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_GUARDED

VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_LAZY

VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_SRCDATASTORE_VALID

VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_DRYRUN

VIXDISKLIB_NASPLUGIN_CLONEFILE_FLAG_SKIPZEROES

4.3 distribute_vixDiskLibPlugin.h File Reference

Data Structures

- struct VixDiskLibPlugin

Defines

- #define VIXDISKLIB_PLUGIN_MAJOR_VERSION 1
- #define VIXDISKLIB_PLUGIN_MINOR_VERSION 0

Typedefs

- typedef VixError(VixDiskLibPluginInit)(VixDiskLibGenericLogFunc ∗log, VixDiskLibGenericLogFunc ∗warn, VixDiskLibGenericLogFunc ∗panic)
- typedef void(VixDiskLibPluginDone)(void)

Enumerations

- enum VixDiskLibPluginType { VIXDISKLIB_PLUGIN_TYPE_TRANSPORT, VIXDISKLIB_PLUGIN_TYPE_NAS, VIXDISKLIB_PLUGIN_TYPE_TRANSPORT_NO_UNLOAD }

Variables

- VixDiskLibPlugin ∗∗ VixDiskLibPlugin_EntryPoint

4.3.1 Define Documentation

4.3.1.1 #define VIXDISKLIB_PLUGIN_MAJOR_VERSION 1

4.3.1.2 #define VIXDISKLIB_PLUGIN_MINOR_VERSION 0

4.3.2 Typedef Documentation

4.3.2.1 typedef VixError(VixDiskLibPluginInit)(VixDiskLibGenericLogFunc ∗log, VixDiskLibGenericLogFunc ∗warn, VixDiskLibGenericLogFunc ∗panic)

Prototype for initializing a plugin in the library. This function will be called when the plugin is loaded to initializze the plugin. If anything but VIX_OK is returned, the plugin will not be loaded.

Parameters:

log IN Log function.

log IN Warning function.

log IN Panic function.

4.3.2.2 typedef void(VixDiskLibPluginDone)(void)

Function to be called when a plugin is unloaded.

4.3.3 Enumeration Type Documentation

4.3.3.1 enum VixDiskLibPluginType

Enumerator:

VIXDISKLIB_PLUGIN_TYPE_TRANSPORT

VIXDISKLIB_PLUGIN_TYPE_NAS

VIXDISKLIB_PLUGIN_TYPE_TRANSPORT_NO_UNLOAD

4.3.4 Variable Documentation

4.3.4.1 VixDiskLibPlugin** VixDiskLibPlugin_EntryPoint

Main entry point into plugin that must be exported. This is an array of pointers to ViDiskLibPlugin structures corresponding to the plugins in this library. The last element in the array must be a NULL pointer.

4.4 public_vm_basic_types.h File Reference

Data Structures

- struct VMPoint
- struct VMRect

Defines

- #define INCLUDE_ALLOW_USERLEVEL
- #define INCLUDE_ALLOW_MODULE
- #define INCLUDE_ALLOW_VMMON
- #define INCLUDE_ALLOW_VMKERNEL
- #define INCLUDE_ALLOW_VMKDRIVERS
- #define INCLUDE_ALLOW_VMK_MODULE
- #define INCLUDE_ALLOW_DISTRIBUTE
- #define INCLUDE_ALLOW_VMCORE
- #define vm_x86_64 0
- #define vm_arm_64 0
- #define vm_64bit (sizeof (void ∗) == 8)
- #define _XTYPEDEF_BOOL
- #define FALSE 0
- #define TRUE 1
- #define IS_BOOL(x) (((x) & ∼1) == 0)
- #define CONST3264(a) (a)
- #define CONST3264U(a) (a)
- #define MIN_INT8 ((int8)0x80)
- #define MAX_INT8 ((int8)0x7f)
- #define MIN_UINT8 ((uint8)0)
- #define MAX_UINT8 ((uint8)0xff)
- #define MIN_INT16 ((int16)0x8000)
- #define MAX_INT16 ((int16)0x7fff)
- #define MIN_UINT16 ((uint16)0)
- #define MAX_UINT16 ((uint16)0xffff)
- #define MIN_INT32 ((int32)0x80000000)
- #define MAX_INT32 ((int32)0x7fffffff)
- #define MIN_UINT32 ((uint32)0)
- #define MAX_UINT32 ((uint32)0xffffffff)
- #define MIN_INT64 (CONST64(0x8000000000000000))
- #define MAX_INT64 (CONST64(0x7fffffffffffffff))
- #define MIN_UINT64 (CONST64U(0))
- #define MAX_UINT64 (CONST64U(0xffffffffffffffff))
- #define AsPercent(v) ((Percent)(v))
- #define UINT64_2_BPN(u) ((BPN)(u))
- #define BPN_2_UINT64(b) ((uint64)(b))
- #define INVALID_WORLD_ID ((World_ID)0)
- #define INVALID_CARTEL_ID INVALID_WORLD_ID
- #define INVALID_SESSION_ID INVALID_CARTEL_ID
- #define INVALID_CARTELGROUP_ID INVALID_CARTEL_ID
- #define INVALID_WORLDLET_ID ((Worldlet_ID)-1)

- #define LA_2_LPN(_la) ((_la) >> PAGE_SHIFT)
- #define LPN_2_LA(_lpn) ((_lpn) << PAGE_SHIFT)
- #define LAST_LPN ((((LA) 1) << (8 ∗ sizeof(LA) - PAGE_SHIFT)) - 1)
- #define LAST_LPN32 ((((LA32)1) << (8 ∗ sizeof(LA32) - PAGE_SHIFT)) - 1)
- #define LAST_LPN64 ((((LA64)1) << (8 ∗ sizeof(LA64) - PAGE_SHIFT)) - 1)
- #define LPN_MASK LAST_LPN
- #define LPN_MASK32 LAST_LPN32
- #define LPN_MASK64 LAST_LPN64
- #define MAX_PPN_BITS 33
- #define MAX_PPN (((PPN)1 << MAX_PPN_BITS) - 1)
- #define INVALID_PPN ((PPN)0x000fffffffffffffull)
- #define INVALID_PPN32 ((PPN32)0xffffffff)
- #define APIC_INVALID_PPN ((PPN)0x000ffffffffffffeull)
- #define INVALID_BPN ((BPN)0x0000ffffffffffffull)
- #define MPN38_MASK ((1ull << 38) - 1)
- #define RESERVED_MPN ((MPN)0)
- #define INVALID_MPN ((MPN)MPN38_MASK)
- #define MEMREF_MPN ((MPN)MPN38_MASK - 1)
- #define RELEASED_MPN ((MPN)MPN38_MASK - 2)
- #define MAX_MPN ((MPN)MPN38_MASK - 3)
- #define INVALID_IOPN ((IOPN)-1)
- #define MAX_IOPN (INVALID_IOPN - 1)
- #define INVALID_LPN ((LPN)-1)
- #define INVALID_VPN ((VPN)-1)
- #define INVALID_LPN64 ((LPN64)-1)
- #define INVALID_PAGENUM ((PageNum)0x000000ffffffffffull)
- #define INVALID_PAGENUM32 ((uint32)-1)
- #define FMTLA ""
- #define FMTVA ""
- #define FMTVPN ""
- #define EXTERN extern
- #define CONST const
- #define INLINE inline
- #define VMX86_EXTERN_DATA extern
- #define INLINE_ALWAYS INLINE
- #define INLINE_SINGLE_CALLER INLINE_ALWAYS
- #define SIDE_EFFECT_FREE
- #define CONST_FUNCTION
- #define NORETURN
- #define HOT
- #define COLD
- #define LIKELY(_exp) (_exp)
- #define UNLIKELY(_exp) (_exp)
- #define PRINTF_DECL(fmtPos, varPos)
- #define SCANF_DECL(fmtPos, varPos)
- #define UNUSED_PARAM(_parm) _parm
- #define UNUSED_TYPE(_parm) UNUSED_PARAM(_parm)
- #define UNUSED_VARIABLE(_var) (void)_var
- #define MUST_CHECK_RETURN
- #define ALIGNED(n)

- #define INFINITE_LOOP() do { } while (1)
- #define FMTPID "d"
- #define FMTUID "u"
- #define FMTMODE "o"

Typedefs

- typedef uint64_t uint64
- typedef int64_t int64
- typedef uint32_t uint32
- typedef int32_t int32
- typedef uint16_t uint16
- typedef int16_t int16
- typedef uint8_t uint8
- typedef int8_t int8
- typedef char Bool
- typedef int64 VmTimeType
- typedef int64 VmTimeRealClock
- typedef int64 VmTimeVirtualClock
- typedef uint8 * TCA
- typedef uint8 Percent
- typedef uintptr_t VA
- typedef uintptr_t VPN
- typedef uint64 PA
- typedef uint64 PPN
- typedef uint64 TPA
- typedef uint64 TPPN
- typedef uint64 PhysMemOff
- typedef uint64 PhysMemSize
- typedef uint64 BA
- typedef uint64 BPN
- typedef uint64 PageCnt
- typedef uint64 PageNum
- typedef unsigned MemHandle
- typedef unsigned IoHandle
- typedef int32 World_ID
- typedef World_ID User_CartelID
- typedef User_CartelID User_SessionID
- typedef User_CartelID User_CartelGroupID
- typedef uint32 Worldlet_ID
- typedef int8 Reg8
- typedef int16 Reg16
- typedef int32 Reg32
- typedef int64 Reg64
- typedef uint8 UReg8
- typedef uint16 UReg16
- typedef uint32 UReg32
- typedef uint64 UReg64
- typedef uint64 MA
- typedef uint32 MPN32

- typedef uint64 SectorType
- typedef uintptr_t LA
- typedef uintptr_t LPN
- typedef uint32 VA32
- typedef uint32 VPN32
- typedef uint32 LA32
- typedef uint32 LPN32
- typedef uint32 PA32
- typedef uint32 PPN32
- typedef uint64 VA64
- typedef uint64 VPN64
- typedef uint64 LA64
- typedef uint64 LPN64
- typedef uint64 PA64
- typedef uint64 PPN64
- typedef uint64 TPPN64
- typedef uint64 MA64
- typedef uint64 MPN
- typedef uint64 IOA
- typedef uint64 IOPN
- typedef VA32 UserVA32
- typedef VA64 UserVA64
- typedef UserVA64 UserVAConst
- typedef UserVA32 UserVA32Const
- typedef UserVA64 UserVA64Const
- typedef void ∗ UserVA
- typedef int64 PollDevHandle
- typedef uint16 utf16_t
- typedef uint32 MX_Rank

4.4.1 Define Documentation

4.4.1.1 #define INCLUDE_ALLOW_USERLEVEL

4.4.1.2 #define INCLUDE_ALLOW_MODULE

4.4.1.3 #define INCLUDE_ALLOW_VMMON

4.4.1.4 #define INCLUDE_ALLOW_VMKERNEL

4.4.1.5 #define INCLUDE_ALLOW_VMKDRIVERS

4.4.1.6 #define INCLUDE_ALLOW_VMK_MODULE

4.4.1.7 #define INCLUDE_ALLOW_DISTRIBUTE

4.4.1.8 #define INCLUDE_ALLOW_VMCORE

4.4.1.9 #define vm_x86_64 0

4.4.1.10 #define vm_arm_64 0

4.4.1.11 #define vm_64bit (sizeof (void ∗) == 8)

4.4.1.12 #define _XTYPEDEF_BOOL

4.4.1.13 #define FALSE 0

4.4.1.14 #define TRUE 1

4.4.1.15 #define IS_BOOL(x) (((x) & ∼1) == 0)

4.4.1.16 #define CONST3264(a) (a)

4.4.1.17 #define CONST3264U(a) (a)

4.4.1.18 #define MIN_INT8 ((int8)0x80)

4.4.1.19 #define MAX_INT8 ((int8)0x7f)

4.4.1.20 #define MIN_UINT8 ((uint8)0)

4.4.1.21 #define MAX_UINT8 ((uint8)0xff)

4.4.1.22 #define MIN_INT16 ((int16)0x8000)

4.4.1.23 #define MAX_INT16 ((int16)0x7fff)

4.4.1.24 #define MIN_UINT16 ((uint16)0)

4.4.1.25 #define MAX_UINT16 ((uint16)0xffff)

4.4.1.26 #define MIN_INT32 ((int32)0x80000000)

4.4.1.27 #define MAX_INT32 ((int32)0x7fffffff)

4.4.1.28 #define MIN_UINT32 ((uint32)0)

4.4.1.29 #define MAX_UINT32 ((uint32)0xffffffff)

4.4.1.30 #define MIN_INT64 (CONST64(0x8000000000000000))

Index