

Xameleon protocols specification

Draft v5

- Note that this document is not a public domain and copyrighted by its author(s) and contributors.
- Note that this document is a draft and subject to change without any prior notification.
- Note that you have to be familiar with the L4 Specification Revision X2 (can be found at <http://l4ka.org/projects/pistachio/l4-x2-r5.pdf>) to understand these protocols.

Document revision history

Revision 1

11.24.2006 Initial document release

Revision 2

11.26.2006 Added new protocols: FileSystem, BlockDevice, CharacterDevice

11.26.2006 Added new data type ThreadID

Revision 3

11.27.2006 Rearranged supervisor's system calls.

11.28.2006 Added protocol layering model

Revision 4

11.30.2006 Modified ReleaseSegment systemcall to return segment's references count

11.30.2006 Added the handle argument to the CreateProcess syscall.

12.01.2006 Added Appendix_I with detail descriptions of a some data types

12.02.2006 Extended semantics of the fork() syscall by a new flag.

12.02.2006 Fixed semantics of the wait() syscall

12.09.2006 Introduced a context argument to the device driver syscalls

01.22.2007 Added ChangeFileTimes syscall

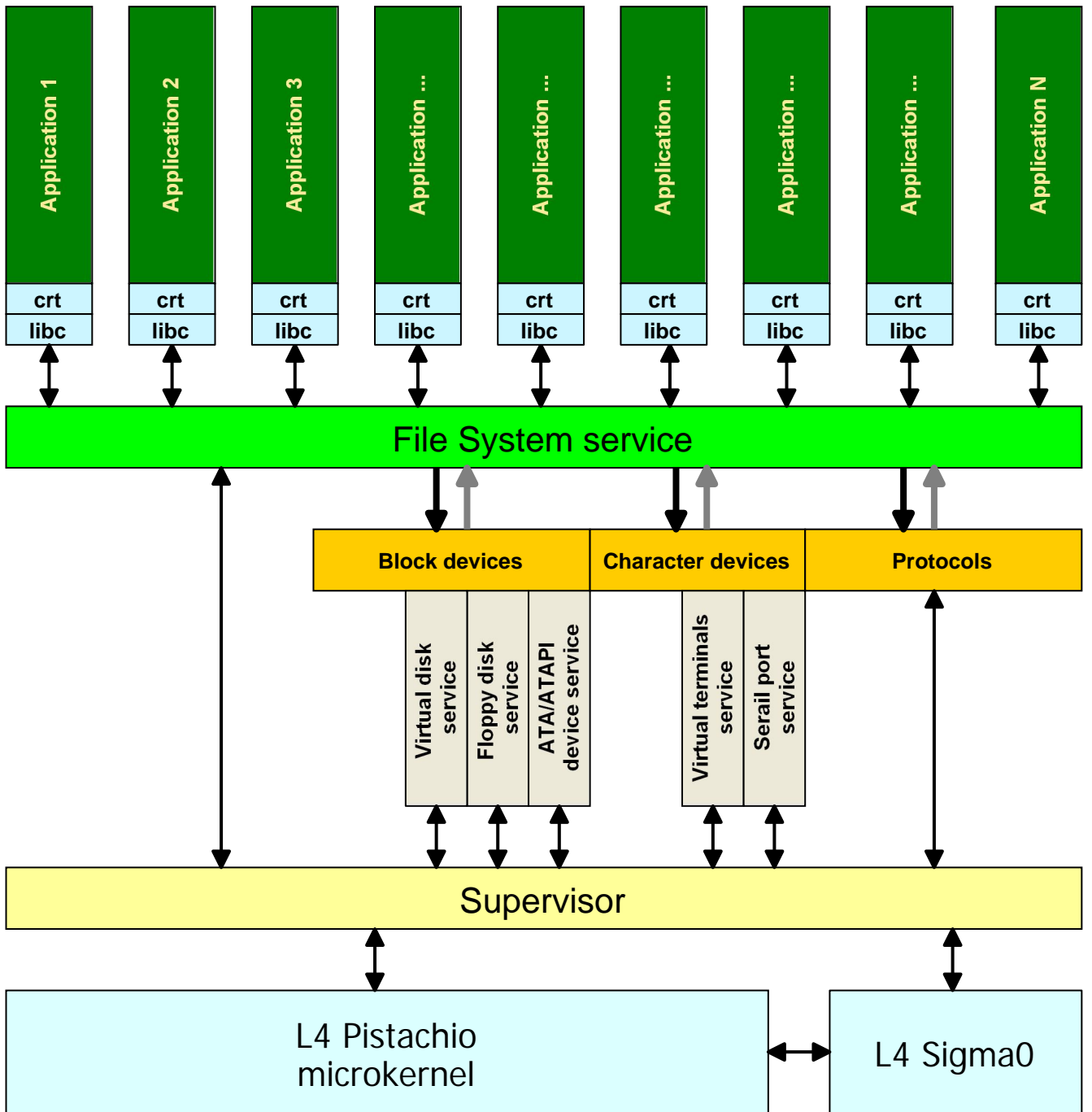
08.24.2007 Change semanitcs of StartDevice syscall

Revision 5

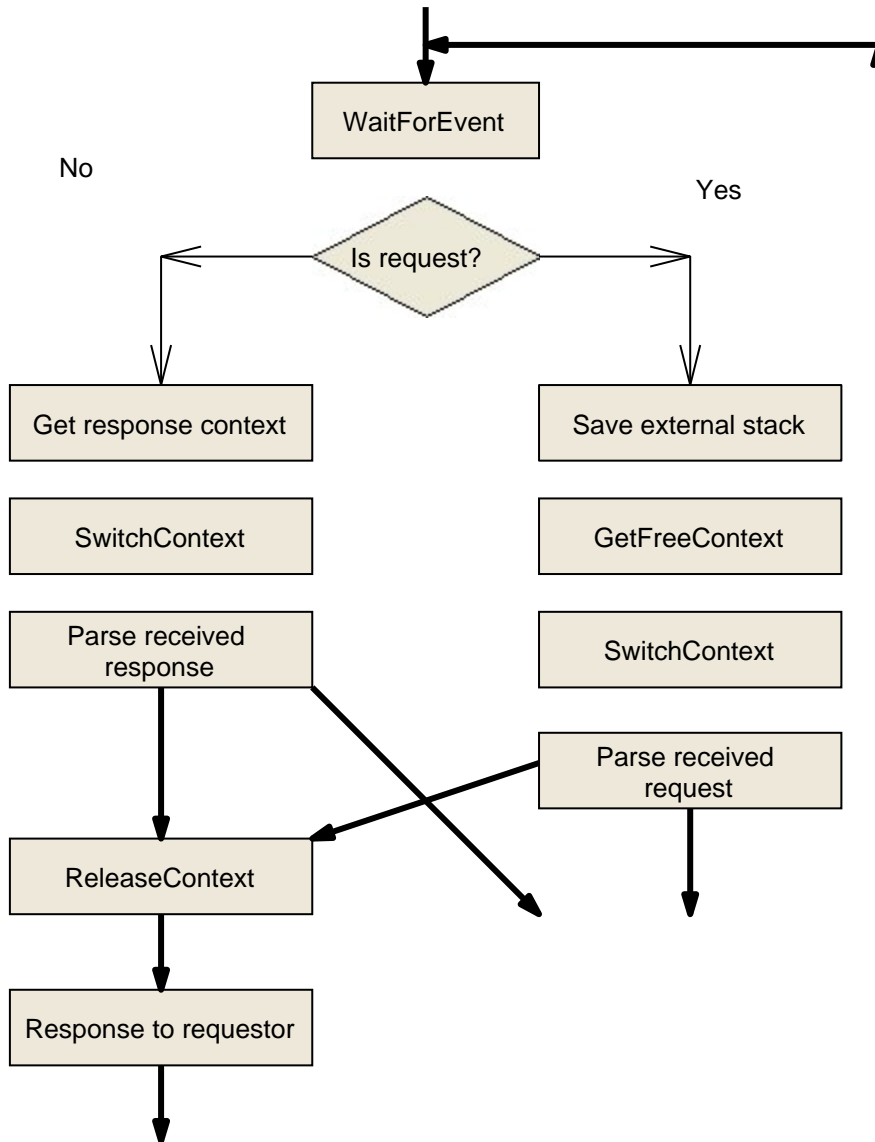
10.11.2007 Add FileSync syscall description into filesystem section

Total 14 entries

Protocol layering model



Preliminary diagram of asynchronous over synchronous syscall implementation



Request to a device driver

Data types information

Following types shown here as a quick hint only.
Refer to L4 X2 specification for detail description of L4 data types.

Type	Description
Word	This data type is the machine word. It's size equal to the architecture word size.
String	The two-word value that represents some memory region within current address space.
Compound	Several pairs of the two-word values that represents several memory regions within current address space. On the IPC receiver side these memory regions share a common message buffer.
MapItem	The two-word value represents L4 flexpage and send base. This item is a part of 'Map as part of message' operation.
ThreadID	The bitfileds that represents programm's thread. This type defined in L4 X2 spec as L4_ThreadId_t. It's size equal to the architecture word size.
Unspecified	This value depends on another argument and out of scope this document.
MsgTag	Message tag bitfiled. This structure identifies message and its content.
DRR	Device registration record as a string or an element of a compound string. This data type conveys information required for a device's startup procedure.
BIH	Xameleon binary image header
DeviceName	Sixteen bytes string that represnts a device name

Supervisor

Supervisor is a roottask in terms of L4 specification. It provides set of methods for creation and destruction processes and threads, managing virtual memory, control device driver and system timers as well as auxiliary functions. Supervisor also starts initial drivers set and process 'init' that arranged in RAM by initial loader.

Method	Description
AllocateSegment	Allocates compound segment. Compound segment represented as set of virtual pages with different sizes that covers requested memory area. These pages will be mapped into caller's address space at address specified by caller in IPC receive phase. For example, 143 Kbytes memory region will be mapped as two pages: 128K + 16K
ReleaseSegment	Releases compound segment that previously has been allocated by AllocateSegment syscall. This operation also unmaps pages that linked to handle from all address space except Supervisor address space. These pages will be used in subsequent AllocateSegment calls. Note that this syscall not warranties that these resources will be freed. In case of several objects share this segment, just a reference count will be decremented and only zero reference count will release segment.
CreateProcess	Runs a new process by its prepared memory image. The process' memory image is covered by a compound segment and must be prepared by other services or modules..
DestroyProcess	Destroys a caller process and all its threads. Releases threads that waiting for caller process. Send program termination status to waiting threads. Note that this request does not provide response do to requestor have to be terminated by this syscall.
CreateThread	Creates a new thread in the caller's address space.
DestroyThread	Destroys thread in the caller's address space. Confirmation sending only in case of requesting thread is not a destroying thread
ForkProcess	Run a new copy of the specified thread as a new process. Note that this syscall is not callable by applications but used by upper layer FileSystem service.
ExecProcess	Executes process' image in a caller's address space. All existing process' threads except leader thread will be destroyed. The Leader thread will be overlapped by executing process. Note that this syscall is not callable by application but used by upper FileSystem service. This syscall rely on a prepared program image that disposed into compound segment specified by the handle argument.
StartDevice	Implements the device starting protocol. All device's parameters transferring via special structure that conveys all device information. This function currently compatible to devices drivers that loaded and parsed into memory by initial boot-loader.
StopDevice	Destroys caller. This method is subject to additional design
DeviceEnumerator	Enumerates registered devices. This method is useful for filesystem driver for creation device filesystem.
GetDeviceHandle	Gets a thread identifier by the device driver name. This function is used by modules to locate each other. Xameleon assumes that the device major number is a L4 thread identifier.
GetProcessID	Returns process ID by its thread ID. Filesystem service uses common resources for all threads within each process. This method provides map between thread identifiers and their process.
GetThreadID	Returns thread ID of a leader thread by his process ID
ChangeHeapSize	Changes size of the caller process' heap. Note that heap starts directly after BSS segment. A new process have a zero size heap prior initialization CRT library. Libc memory functions take memory from the heap by calling this syscall.
SetSignal	Sets signal handling routine for the caller process. New signal handler do not affect to another processes that are not in the caller's address space.
Signal	Sends signal to the process
ProcessWait	Waits for termination of an another process.
SetTimer	Controls timers of the caller process.
SetNotificationHandler	Set a listener thread for various supervisor events
SystemHalt	Does nothing at this time
LeaveSignal	Leave a libc signal handling code.

AllocateSegment

Allocates compound segment. Compound segment represented as set of virtual pages with different sizes that covers requested memory area. These pages will be mapped into caller's address space at address specified by caller in IPC receive phase. For example, 143 Kbytes memory region will be mapped as two pages: 128K + 16K

Message register	Type	Label
Tag	MsgTag	1

Input message register values

Message registers	Type	Name	Description
MR1	Word	address	Virtual address of requested segment
MR2	Word	size	Size of requested segment

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	Word	handle	Handle to allocated segment
MR3	Word	pages_count	Pages count to be mapped. Note that we using compound pages. That means that pages is an array of different size pages, which covers requested memory region.
MR4...n	MapItem	pages	Pages array. See description of the MapItem and mapping procedure in L4 X2 specification.

ReleaseSegment

Releases compound segment that previously has been allocated by AllocateSegment syscall. This operation also unmaps pages that linked to handle from all address space except Supervisor address space. These pages will be used in subsequent AllocateSegment calls. Notr that this syscall not warranties that these resources will be freed. In case of several objects share this segment, just a reference count will be decremented and only zero reference count will release segment.

Message register	Type	Label
Tag	MsgTag	2

<i>Input message register values</i>			
Message registers	Type	Name	Description
MR1	Word	handle	Handle to releasing segment

<i>Output message register values</i>			
Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	Word	ref_count	References count to this segment

CreateProcess

Runs a new process by its prepared memory image. The process' memory image is covered by a compound segment and must be prepared by other services or modules..

Message register	Type	Label
Tag	MsgTag	3

Input message register values

Message registers	Type	Name	Description
MR 1	Word	handle	Handle to a compound segment where the process image resides
MR 2	Word	entry_point	Entry point to the process
MR 3	Word	text_vstart	Virtual address of the text segment
MR 4	Word	text_size	Size of text segment
MR 5	Word	data_vstart	Virtual address the data segment
MR 6	Word	data_size	Size of the text segment
MR 7	Word	bss_vstart	Virtual address of the BSS segment
MR 8	Word	bss_size	Size of the BSS segment
MR 9,10	Compound	argv	String that represents process' arguments
MR11,12	Compound	env	String that represents process' environment

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	ThreadID	thread_id	Thread identifier of newly created process or nilthread if process creation fails
MR3	Word	pid	Process ID of the newly cretaed process

DestroyProcess

Destroys a caller process and all its threads. Releases threads that waiting for caller process. Send program termination status to waiting threads. Note that this request does not provide response do to requestor have to be terminated by this syscal.

Message register	Type	Label
Tag	MsgTag	4

Input message register values

Message registers	Type	Name	Description
MR1	Word	status	Program termination status

CreateThread

Creates a new thread in the caller's address space.

Message register	Type	Label
Tag	MsgTag	5

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	Word	ip	Entry point to the thread
MR2	Word	stack_bottom	Bottom address of the stack
MR3	Word	stack_size	Size of stack

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	ThreadID	thread_id	Thread identifier of a newly created thread or nilthread if process creation fails

DestroyThread

Destroys thread in the caller's address space. Confirmation sending only in case of requesting thread is not a destroying thread

Message register	Type	Label
Tag	MsgTag	6

Input message register values

Message registers	Type	Name	Description
MR1	ThreadID	thread_id	Thread identifier of destroying thread

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status

ForkProcess

Run a new copy of the specified thread as a new process. Note that this syscall is not callable by applications but used by upper layer FileSystem service.

Message register	Type	Label
Tag	MsgTag	7

Input message register values

Message registers	Type	Name	Description
MR1	ThreadID	thread_id	Thread identifier of the forking thread

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	ThreadID	thread_id	Thread identifier of forked process or nilthread if fork is failed
MR3	Word	process_id	Process identifier (pid_t) of newly created process

ExecProcess

Executes process' image in a caller's address space. All existing process' threads except leader thread will be destroyed. The Lader thread will be overlapped by executing process. Not that this syscall is not callable by application but used by upper FileSystem service. This syscall rely on a prepared program image that disposed into compound segment specified by the handle argument.

Message register	Type	Label
Tag	MsgTag	8

Input message register values

Message registers	Type	Name	Description
MR 1	Word	thread_id	Thread identifier that describes an overlapping thread
MR 2	Word	handle	Handle to a compound memory segment that covers all sections of the executable image.
MR 3	Word	entry_point	Entry point of the executable image.
MR 4	Word	text_start	Virtual address of text sgment
MR 5	Word	text_size	Size of text segment
MR 6	Word	data_start	Virtual address of data segment
MR 7	Word	data_size	Size of data segment
MR 8	Word	bss_start	Virtual address of BSS segment
MR 9	Word	bss_size	Size of BSS segment
MR10,11	Compound	arguments	String that represent programm calling arguments
MR12,13	Compound	environment	String that represent process environment

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status. Note that this value is used by filesystem driver

StartDevice

Implements the device starting protocol. All device's parameters transferring via special structure that conveys all device information. This function currently compatible to devices drivers that loaded and parsed into memory by initial bootloader.

Message register	Type	Label
Tag	MsgTag	9

Input message register values

Message registers	Type	Name	Description
MR1	Word	space	0 - run in driver recommended address space, 1 - run in Supervisor's address space, 2 - run in dedicated address space
MR2,3	String	device_record	The string that represents a device registration record

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	ThreadID		ThreadID of a main therad of the driver

StopDevice

Destroys caller. This method is subject to additional design

Message register	Type	Label
Tag	MsgTag	10

Input message register values

Message registers	Type	Name	Description
MR1	Unspecified	...	Format is not designed yet

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status

DeviceEnumerator

Enumerates registered devices. This method is useful for filesystem driver for creation device filesystem.

Message register	Type	Label
Tag	MsgTag	11

Input message register values

Message registers	Type	Name	Description
MR1	Word	device_no	Device number. Increment it until successful status

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status code
MR2	Word	total	Total device count
MR3,4	String	dev_info	Public device record

GetDeviceHandle

Gets a thread identifier by the device driver name. This function is used by modules to locate each other. Xameleon assumes that the device major number is a L4 thread identifier.

Message register	Type	Label
Tag	MsgTag	12

Input message register values

Message registers	Type	Name	Description
MR1,2	String	device_name	Name of requested device driver

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	ThreadID	thread_id	Thread identifier of device driver or nilthread if device does not exist

GetProcessID

Returns process ID by its thread ID. Filesystem service uses common resources for all threads within each process. This method provides map between thread identifiers and their process.

Message register	Type	Label
Tag	MsgTag	13

Input message register values

Message registers	Type	Name	Description
MR1	ThreadID	thread_id	Thread identifier of process that ID is requested

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	Word	process_id	Process id for thread_id argument. See POSIX pid_t
MR3	Word	parent_id	Parent process id for thread argument. See POSIX ppid_t

GetThreadID

Returns thread ID of a leader thread by his process ID

Message register	Type	Label
Tag	MsgTag	14

Input message register values

Message registers	Type	Name	Description
MR1	Word	process_id	Process identifier

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2	ThreadID	thread_id	Main thread identifier of requested process

ChangeHeapSize

Changes size of the caller process' heap. Note that heap starts directly after BSS segment. A new process have a zero size heap prior initialization CRT library. Libc memory functions take memory from the heap by calling this syscall.

Message register	Type	Label
Tag	MsgTag	15

Input message register values

Message registers	Type	Name	Description
MR1	Word	increment	Signed value that indicates heap increment number bytes

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Operation status code.
MR2	Word	bottom	Heap start virtual address
MR3	Word	old_top	Heap top virtual address
MR4	Word	new_top	Old heap top value

SetSignal

Sets signal handling routine for the caller process. New signal handler do not affect to another processes that are not in the caller's address space.

Message register	Type	Label
Tag	MsgTag	16

Input message register values

Message registers	Type	Name	Description
MR1	Word	signal_number	Signal number
MR2	String	handler	String that represents the POSIX sigaction structure

Output message register values

Message registers	Type	Name	Description
MR1	Word	status	Status of setting signal handler operation
MR2	String	old_handler	String that represents the POSIX sigaction structure that describes previous settings

Signal

Sends signal to the process

Message register	Type	Label
Tag	MsgTag	17

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	ThreadID	thread_id	Thread identifier of destination thread
MR2	Word	signal_number	Signal number

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	Word	status	Signal delivery status
MR2	Word	reserver	Reserved value

ProcessWait

Waits for termination of an another process.

Message register	Type	Label
Tag	MsgTag	18

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	ThreadID	thread_id	Thread identifier of waitnig process or anythread for waiting any child thread.
MR2	Word	options	Waiting options. See POSIX wait

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	Word	status	Status of wait operation
MR2	Word	exit_status	Exit status of terminated process or signal number
MR3	ThreadID	thread_id	Thread identifier of terminated process
MR4	Word	process_id	Process identifier of terminated process

SetTimer

Controls timers of the caller process.

Message register	Type	Label
Tag	MsgTag	19

<i>Input message register values</i>			
Message registers	Type	Name	Description
MR1	Word	timer_id	Timer identifier
MR2,3	String	itimerval	POSIX itimerval structure. Empty string resets timer timer_id.

<i>Output message register values</i>			
Message registers	Type	Name	Description
MR1	Word	status	Operation status
MR2,3	String	itimerval	Previously timer values as POSIX ittimerrval structure

SetNotificationHandler

Set a listener thread for various supervisor events

Message register	Type	Label
Tag	MsgTag	20

Input message register values

Message registers	Type	Name	Description
MR1	Word	EventID	Value reserved for future use (must be 0)
MR2	ThreadID	ThreadID	An event handling thread identifier

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Status of operation

SystemHalt

Does nothing at this time

Message register	Type	Label
Tag	MsgTag	21

LeaveSignal

Leave a libc signal handling code.

Message register	Type	Label
Tag	MsgTag	22

FileSystem

File system provides set of methods for control files, directories and other primitives such as pipes and sockets. (Sockets did not implemented yet)

Method	Description
Mount	Mount filesystem into directory
Unmount	Umount mounted filesystem
GetCurrentDirectory	Get current directory of the calling process
ChangeDir	Change curent directory of the calling process
OpenDirectory	Open directory for read
ReadDirectory	Read directory entry
CloseDirectory	Close directory
GetFileStatus	Get file status
ChangeRootDirectory	Change root directory (i.e. Jail)
OpenFile	Open/create a regular file
CloseFile	Close file
ReadFile	Read file
WriteFile	Write file
RemoveFile	Remove file
RemoveDirectory	Remove directory
CreateDirectory	Create directory
CreateHardLink	Create hard link
CreateSybolicLink	Create symbolic link
RenameFile	Rename file
SeekFile	Seek a file pointer
GetSuperblockStatus	Get superblock status
ChangeOwner	Change file owner
ChangeMode	Change file mode
FlushCaches	Flush disk caches
DupDescriptor	Duplicate file descriptor
CopyDescriptor	Copy file descriptor
CreatePipe	Create pipe
ForkProcess	Fork process
Exec	Exec process
MakeNode	Make node
IOCTL	IOCTL
Exit	Exit process with status
ControlFile	POSIX fcntl function
ChaneFileTimes	This syscall emulates an utimes syscall
CreateSession	This syscall emulates a POSIX sesid syscall
FileSync	Flush cashes and other filesystem structures related to a file descriptor to a physical device

Mount

Mount filesystem into directory

Message register	Type	Label
Tag	MsgTag	64

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR0	Word	Flags	Mount flags
MR1,2	Compound	Device	String that represents the device name to be mounted
MR3,4	Compound	Point	String that represent the mount point within a filesystem tree
MR5,6	Compound	Type	Filesystem type

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR0	Word	Status	Mount operation status

Unmount

Unmount mounted filesystem

Message register	Type	Label
Tag	MsgTag	65

Input message register values

Message registers	Type	Name	Description
MR0,1	String	Point	The mount point

Output message register values

Message registers	Type	Name	Description
MR0	Word	Status	Operation status

GetCurrentDirectory

Get current directory of the calling process

Message register	Type	Label
Tag	MsgTag	66

Input message register values

Message registers	Type	Name	Description
MR1	Word	Size	Receiver's buffer size.

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2,3	String	Path	String that represent current process' path.

ChangeDir

Change curent directory of the calling process

Message register	Type	Label
Tag	MsgTag	67

Input message register values

Message registers	Type	Name	Description
MR1,2	String	Path	New process path

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

OpenDirectory

Open directory for read

Message register	Type	Label
Tag	MsgTag	68

Input message register values

Message registers	Type	Name	Description
MR1,2	String	Path	Path to directory for open

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2	Word	Handle	Directory handle.

ReadDirectory

Read directory entry

Message register	Type	Label
Tag	MsgTag	69

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	Handle to directory

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Status of operation
MR2,3	String	Record	Directory record. Subsequent calls return next directory entry

CloseDirectory

Close directory

Message register	Type	Label
Tag	MsgTag	70

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	Handle to directory

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

GetFileStatus

Get file status

Message register	Type	Label
Tag	MsgTag	71

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	Word	Parameters	0-follow symlinks, 1-allow symlink, any other value is open file descriptor
MR2,3	String	FileName	Name of requested file

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2,3	String	Attributes	File attributes

ChangeRootDirectory

Change root directory (i.e. Jail)

Message register	Type	Label
Tag	MsgTag	72

Input message register values

Message registers	Type	Name	Description
MR1,2	String	Path	Path to new root direcorey of caller process

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Opeartion status

OpenFile

Open/create a regular file

Message register	Type	Label
Tag	MsgTag	73

<i>Input message register values</i>			
Message registers	Type	Name	Description
MR1	Word	Flags	File open flags. Flags compatible with POSIX open() flags
MR2	Word	Mode	File open mode. Mode compatible with POSIX file modes.
MR3,4	String	Filename	The filename of file to open/create

<i>Output message register values</i>			
Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2	Word	Handle	Positive value is file handle, negative value is error code

CloseFile

Close file

Message register	Type	Label
Tag	MsgTag	74

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File handle to close

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operaion status

ReadFile

Read file

Message register	Type	Label
Tag	MsgTag	75

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File handle to read
MR2	Word	BytesCount	Count bytes to read

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Read status. Positive value is bytes read count, negative is error code
MR2,3	String	Data	Data that have been read from file

WriteFile

Write file

Message register	Type	Label
Tag	MsgTag	76

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File handle to write
MR2,3	String	Data	Data for writing to file

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Write status. Positive value is bytes write count, negative is error code

RemoveFile

Remove file

Message register	Type	Label
Tag	MsgTag	77

Input message register values

Message registers	Type	Name	Description
MR1,2	String	Filename	The name of file to delete

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Deletion status

RemoveDirectory

Remove directory

Message register	Type	Label
Tag	MsgTag	78

Input message register values

Message registers	Type	Name	Description
MR1,2	String	DirName	The directory name to delete

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

CreateDirectory

Create directory

Message register	Type	Label
Tag	MsgTag	79

Input message register values

Message registers	Type	Name	Description
MR1	Word	Mode	Creation mode
MR2,3	String	DirName	The new directory name

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

CreateHardLink

Create hard link

Message register	Type	Label
Tag	MsgTag	80

Input message register values

Message registers	Type	Name	Description
MR1,2	Compound	Source	Source filename
MR3,4	Compound	Target	Target filename

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

CreateSybolicLink

Create symbolic link

Message register	Type	Label
Tag	MsgTag	81

Input message register values

Message registers	Type	Name	Description
MR1,2	Compound	Source	Source filename
MR3,4	Compound	Target	Target filename

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

RenameFile

Rename file

Message register	Type	Label
Tag	MsgTag	82

Input message register values

Message registers	Type	Name	Description
MR1,2	Compound	OldName	File name to change
MR3,4	MapItem	NewName	New name of the file

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

SeekFile

Seek a file pointer

Message register	Type	Label
Tag	MsgTag	83

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File handle
MR2	Word	Offset	Seek offset (signed)
MR3	Word	Whence	Seek direction

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2	Word	Position	New position within file

GetSuperblockStatus

Get superblock status

Message register	Type	Label
Tag	MsgTag	84

Input message register values

Message registers	Type	Name	Description
MR1,2	String	Path	Path

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2,3	String	Record	Superblock information record

ChangeOwner

Change file owner

Message register	Type	Label
Tag	MsgTag	85

Input message register values

Message registers	Type	Name	Description
MR1	Word	Owner	New file owner
MR2	Word	Group	New file group
MR3,4	String	Filename	Filename

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

ChangeMode

Change file mode

Message register	Type	Label
Tag	MsgTag	86

Input message register values

Message registers	Type	Name	Description
MR1	Word	Mode	File mode
MR2,3	String	Filename	Filename

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

FlushCaches

Flush disk caches

Message register	Type	Label
Tag	MsgTag	87

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

DupDescriptor

Duplicate file descriptor

Message register	Type	Label
Tag	MsgTag	88

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File handle for duplication

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status.
MR2	Word	NewHandle	New descriptor that points to the same file as the input handle

CopyDescriptor

Copy file descriptor

Message register	Type	Label
Tag	MsgTag	89

Input message register values

Message registers	Type	Name	Description
MR1	Word	Target	Target file descriptor
MR2	Word	Source	Source file descriptor

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

CreatePipe

Create pipe

Message register	Type	Label
Tag	MsgTag	90

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status
MR2	Word	Pipeln	Pipe input endpoint
MR3	Word	PipeOut	Pipe output endpoint

ForkProcess

Fork process

Message register	Type	Label
Tag	MsgTag	91

<i>Input message register values</i>			
Message registers	Type	Name	Description
MR1	Word	Flags	These flags describes a fork() extension modes. Provide zero value to follow the POSIX behaviour

<i>Output message register values</i>			
Message registers	Type	Name	Description
MR1	Word	ProcessID	Process identifier of newly created child process
MR2	ThreadID	ThreadID	Thread identifier of new created process

Exec

Exec process

Message register	Type	Label
Tag	MsgTag	92

Input message register values

Message registers	Type	Name	Description
MR1,2	Compound	ProcessPath	Path to executable file
MR3,4	Compound	Arguments	Programm command line arguments

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Exec status. Status is never returned on success execution.

MakeNode

Make node

Message register	Type	Label
Tag	MsgTag	96

Input message register values

Message registers	Type	Name	Description
MR0	Word	Mode	File mode
MR1	ThreadID	Major	Major device number
MR2	Word	Minor	Minor devcie number
MR3	Word	Size	Object size
MR4,5	String	Name	Path and name new node

Output message register values

Message registers	Type	Name	Description
MR0	Word	Status	Operation status

IOCTL

IOCTL

Message register	Type	Label
Tag	MsgTag	97

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File descriptor
MR2	Word	Command	Control command ID
MR3...	Unspecified	...	Format of these registers depends on Command type

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Control command execution status
MR2...	Unspecified	Format of these registers depends on Command type

Exit

Exit process with status

Message register	Type	Label
Tag	MsgTag	98

<i>Input message register values</i>			
Message registers	Type	Name	Description
MR1	Word	Status	Status that must be sent to waiting processes

ControlFile

POSIX fcntl function

Message register	Type	Label
Tag	MsgTag	99

<i>Input message register values</i>			
Message registers	Type	Name	Description
MR1	Word	Handle	File handle to control
MR2	Word	Mode	Control mode
MR3...	Unspecified	...	Format of these registers depends on command mode

<i>Output message register values</i>			
Message registers	Type	Name	Description
MR1	Word	Status	Control operation status
MR2...	Unspecified	...	Format of these registers depends on Command mode

ChaneFileTimes

This syscall emulates an utimes syscall

Message register	Type	Label
Tag	MsgTag	100

Input message register values

Message registers	Type	Name	Description
MR1,2	String	FileName	The name of file which times will be set by following structure
MR3,4	String	FileTimes	This string conveys an utimbuf_t structure

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Operation status

CreateSession

This syscall emulates a POSIX sesid syscall

Message register	Type	Label
Tag	MsgTag	101

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Status of operation
MR2	Word	ProcessID	pid_t of new session

FileSync

Flush caches and other filesystem structures related to a file descriptor to a physical device

Message register	Type	Label
Tag	MsgTag	102

Input message register values

Message registers	Type	Name	Description
MR1	Word	Handle	File handle to flush it's caches

Output message register values

Message registers	Type	Name	Description
MR1	Word	Status	Status of operation

BlockDevice

This protocol describes methods for controlling block devices, such as hard drives, CDROMs, flash and remote filesystem devices.

Method	Description
BlockDevice_Open	Get type of the disk partition, virtual or generic drive
BlockDevice_Flush	Flush disk caches. Requestor does not rely on caches type - software or hardware.
BlockDevice_Read	Read device block by its number
BlockDevice_Write	Write device block by its number
BlockDevice_IOCTL	Control block device: power modes, DMA modes, caching modes, remote filesystem options, and so on.

BlockDevice_Open

Get type of the disk partition, virtual or generic drive

Message register	Type	Label
Tag	MsgTag	1

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Minor	Minor device number
MR2	ThreadID	Listenr	An ID of the response listener thread

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Status	Operation status
MR2	Word	ID	Filesystem ID. This value conforms to MBR .
MR3	Word	SectorSize	Device's sector size
MR4	Word	FirstSector	First sector number of partition/session relative to the first sector of a physical media

BlockDevice_Flush

Flush disk caches. Requestor does not rely on caches type - software or hardware.

Message register	Type	Label
Tag	MsgTag	2

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Minor	Minor device number
MR2	Word	reserved	Do we need flush caches of special type? Like inodes blocks, file data bocks, inodes bitmap blocks, free block bitmap blocks, system area? If so, then this argument would be nice for cache type description.

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Status	Operation status

BlockDevice_Read

Read device block by its number

Message register	Type	Label
Tag	MsgTag	3

<i>Input message register values</i>			
--------------------------------------	--	--	--

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Minor	Minor device number
MR2	Word	BlockNumber	Identifies requested block number
MR3	Word	BlockSize	Identifies block size
MR4	Word	CachingType	Determines caching algorithm for requested block

<i>Output message register values</i>			
---------------------------------------	--	--	--

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Status	Operation status
MR2,3	String	Data	Requested data

BlockDevice_Write

Write device block by its number

Message register	Type	Label
Tag	MsgTag	4

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Minor	Minor device number
MR2	Word	BlockNumber	Identifies writing block number
MR3	Word	BlockSize	Identifies block size
MR4	Word	CachingType	Describes data type on top of that a caching scheme is selecting by the device driver
MR5,6	String	Data	Writing data

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Status	Operation status

BlockDevice_IOCTL

Control block device: power modes, DMA modes, caching modes, remote filesystem options, and so on.

Message register	Type	Label
Tag	MsgTag	5

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Minor	Minor device number
MR2	Word	Command	Control command ID
MR3...	Unspecified	...	Format of these registers depends on Command type

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	Context
MR1	Word	Status	Control command execution status
MR2...	Unspecified	...	Format of these registers depends on Command type

CharacterDevice

Character devices protocol describes methods for controlling character devices, such as terminals, serial ports and so on.

Method	Description
StreamDevice_Open	Open a stream device
StreamDevice_Close	Close a stream device that previously was opened by StreamDevice_Open syscall
StreamDevice_Read	Read bytes/packets/strings from a stream device
StreamDevice_Write	Writes bytes/packets/strings to a stream device
StreamDevice_IOCTL	Control stream device

StreamDevice_Open

Open a stream device

Message register	Type	Label
Tag	MsgTag	1

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Requestor's context, this value must be returned within response and it unique identifies response
MR1	Word	Minor	Minor device number
MR2	ThreadID	Listener	An ID of the response listener thread
MR3	Word	Flags	Flags for method open()

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	This argument must provide the same value as the context argument of the input message
MR1	Word	Status	Operation status

StreamDevice_Close

Close a stream device that previously was opened by StreamDevice_Open syscall

Message register	Type	Label
Tag	MsgTag	2

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Requestor's context, this value must be returned within response and it unique identifies response
MR1	Word	Minor	Minor device number

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	This argument must provide the same value as the context argument of the input message
MR1	Word	Status	Operation status

StreamDevice_Read

Read bytes/packets/strings from a stream device

Message register	Type	Label
Tag	MsgTag	3

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Requestor's context, this value must be returned within response and it unique identifies response
MR1	Word	Minor	Minor device number
MR2	Word	Size	Maximum bytes to read.

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	This argument must provide the same value as the context argument of the input message
MR1	Word	Status	Count bytes have been read from device. Negative value signalling error and error code
MR2,3	String	Data	Data that have been read from device. This data will be copyied into receiver's buffer

StreamDevice_Write

Writes bytes/packets/strings to a stream device

Message register	Type	Label
Tag	MsgTag	4

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Requestor's context, this value must be returned within response and it unique identifies response
MR1	Word	Minor	Minor device number
MR2,3	String	Data	String that represents data for writing to device

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	This argument must provide the same value as the context argument of the input message
MR1	Word	Status	Count bytes that have been witten to device. Negative value signaling error and error code

StreamDevice_IOCTL

Control stream device

Message register	Type	Label
Tag	MsgTag	5

Input message register values

Message registers	Type	Name	Description
MR0	Word	Context	Requestor's context, this value must be returned within response and it unique identifies response
MR1	Word	Minor	Minor device number
MR2	Word	Command	Control command ID
MR3...	Unspecified	...	Format of these registers depends on Command type

Output message register values

Message registers	Type	Name	Description
MR0	Word	Context	This argument must provide the same value as the context argument of the input message
MR1	Word	Status	Control command execution status
MR2...	Unspecified	...	Format of these registers depends on Command type

Device registration record as a string or an element of a compound string. This data type conveys information required for a device's startup procedure.

```
typedef struct DRR {
    Word      ProtocolVersion;    // Version of the startup protocol
    Word      Category;           // Driver category: 0 - RAW, 1 - Stream, 2 - Block, 3 -
                                // Protocol
    Word      Model;              // Driver execution model: 0 - Module, 1 - Process, 2 -
                                // Active thread
    Word      FirstMinor;         // First minor number of a device set provided by the
                                // driver
    Word      MinorDeviceCount;   // Minor device count
    BIH       ImageHeader;        // Binary image header
    Word      MaxHeapSize;        // Maximum size of the driver's heap
    DeviceName DeviceName;        // Device name that sent this request
    Word      MagicNumber;        // This field must convey hex value 0xfeedface
} DRR;
```

Xameleon binary image header

```
typedef struct BIH {
    Word      EntryPoint;         // entry point to the code
    Word      TextStart;          // start address of the code segment. Note that a
                                // current implementation does not distinguish between code
                                // and read-only data sections,
    Word      TextSize;           // size of the code segment
    Word      DataStart;          // start address of the initialized writable data
                                // segment
    Word      DataSize;           // size of the the initialized writable data segment
    Word      BssStart;           // start address of the BSS
    Word      BssSize;            // size of the BSS
    Word      StackBottom;        // Semantics of this field depends on StackSize
                                // argument. If stack size is not zero, this value
                                // describes the bottom stack address. In other case, this
                                // value describes a maximum limit of executable stack
    Word      StackSize;          // Zero value forces a system stack allocation policy.
                                // Any other value describes programm stack size.
} BIH;
```