

Security Research

PASR

Preparatory Action on the enhancement of the European industrial potential in the field of Security research



Grant Agreement no. SEC5-PR-104600
ROBIN

Open Robust Infrastructures

Project

Deliverable D.4

Informal Specification of Key TCB Components

Due date of deliverable: 31/04/2008
Actual submission date: 15/05/2008

Start date of Activity: 31/01/2006

Duration: 31/04/2008

Organisation name of lead beneficiary for this deliverable: Technische Universität Dresden

Revision 2

CLASSIFICATION: Public

15/05/2008

Specification of the Bastei OS Architecture

Norman Feske and Christian Helmuth

March 27, 2008

1 Bastei OS Architecture Namespace Documentation

1.1 Bastei Namespace Reference

CPU (processing time) manager session interface.

Classes

- class [Allocator](#)
- class [Allocator_avl_base](#)
- class [Allocator_avl_tpl](#)
- class [Empty](#)
- class [Session](#)
- class [Child](#)
- class [Env](#)
- class [Heap](#)
- class [Sliced_heap](#)
- class [Ipc_error](#)
- class [Buffer](#)
- class [Ipc_marshall](#)
- class [Ipc_unmarshaller](#)
- class [Ipc_ostream](#)
- class [Ipc_istream](#)
- class [Ipc_client](#)
- class [Ipc_server](#)
- class [Msgbuf](#)
- class [Lock](#)
- class [Lock_guard](#)
- class [Object_pool](#)
- class [Ram_dataspace](#)
- class [Semaphore](#)
- class [Server_object](#)
- class [Server_activation_base](#)

- class [Server_entrypoint](#)
- class [Server_activation](#)
- class [Client](#)
- class [Service](#)
- class [Local_service](#)
- class [Remote_service](#)
- class [Service_pool](#)
- class [Remote_service_pool](#)
- class [Session_control](#)
- class [Slab_block](#)
- class [Slab_entry](#)
- class [Slab](#)
- class [Task](#)
- class [Thread_base](#)
- class [Thread](#)
- struct [Thread_state](#)
- class [Tslab](#)
- class [Cap_session](#)
- class [Cpu_session](#)
- class [Dataspace](#)
- class [Io_mem_session](#)
- class [Io_port_session](#)
- class [Irq_session](#)
- class [Log_session](#)
- class [Parent](#)
- class [Ram_session](#)
- class [Rm_session](#)
- class [Rom_session](#)
- class [Root](#)
- class [Task_session](#)

Typedefs

- typedef [Allocator_avl_tpl](#)< [Empty](#) > [Allocator_avl](#)
- typedef [Fiasco::Capability](#) [Capability](#)
- typedef [Fiasco::l4_threadid_t](#) [Connection_state](#)
- typedef [Fiasco::Msgbuf_base](#) [Msgbuf_base](#)
- typedef [Service_pool](#)< [Local_service](#) > [Local_service_pool](#)

Enumerations

- enum { [ERR_INVALID_OBJECT](#) = -1 }
- enum [Ipc_ostream_send](#) { [IPC_SEND](#) }
- enum [Ipc_istream_wait](#) { [IPC_WAIT](#) }
- enum [Ipc_client_call](#) { [IPC_CALL](#) }
- enum [Ipc_server_reply_wait](#) { [IPC_REPLY_WAIT](#) }

Functions

- `template<typename T>`
void `destroy` (`Allocator *alloc`, `T *obj`)
- `Env * env` ()
- `Capability parent_cap` ()
- `Capability session` (`const char *service_name`, `const char *format_args`,...)
- void `sleep_forever` ()

1.1.1 Detailed Description

`Task` (program) manager session interface.

Author:

Christian Helmuth

Date:

2006-06-27

:Question:

Why are thread operations not methods of the thread but methods of the CPU session?

:Answer:

This enables the CPU session to impose policies on thread operations. These policies are based on the session construction arguments. If thread operations would be provided as thread methods, `Thread` would need to consult its container object (its CPU session) about the authorization of each operation and, thereby, would introduce a circular dependency between CPU session and `Thread`.

Author:

Christian Helmuth

Date:

2006-06-27

A task session represents the protection domain of a program.

1.1.2 Typedef Documentation

1.1.2.1 typedef `Allocator_avl_tpl<Empty>` `Bastei::Allocator_avl`

1.1.2.2 typedef `::Fiasco::Capability` `Bastei::Capability`

1.1.2.3 typedef `::Fiasco::l4_threadid_t` `Bastei::Connection_state`

1.1.2.4 typedef `Service_pool<Local_service>` `Bastei::Local_service_pool`

1.1.2.5 typedef `::Fiasco::Msgbuf_base` `Bastei::Msgbuf_base`

1.1.3 Enumeration Type Documentation

1.1.3.1 anonymous enum

Enumerator:

ERR_INVALID_OBJECT

1.1.3.2 enum Bastei::Ipc_client_call

Enumerator:

IPC_CALL

1.1.3.3 enum Bastei::Ipc_istream_wait

Enumerator:

IPC_WAIT

1.1.3.4 enum Bastei::Ipc_ostream_send

Enumerator:

IPC_SEND

1.1.3.5 enum Bastei::Ipc_server_reply_wait

Enumerator:

IPC_REPLY_WAIT

1.1.4 Function Documentation

1.1.4.1 `template<typename T> void Bastei::destroy (Allocator * alloc, T * obj)` [inline]

1.1.4.2 `Env* Bastei::env ()`

1.1.4.3 `Capability Bastei::parent_cap ()`

Return parent capability

Platforms have to implement this function for environment initialization.

1.1.4.4 `Capability Bastei::session (const char * service_name, const char * format_args, ...)` [inline]

Shortcut for `env()->parent()->session()` function

1.1.4.5 void Bastei::sleep_forever () [inline]

1.2 Fiasco Namespace Reference

Classes

- class [Capability](#)
- class [Msgbuf_base](#)
- class [Msgbuf](#)

1.3 Framebuffer Namespace Reference

Classes

- class [Session](#)

1.4 Init Namespace Reference

Classes

- class [Child](#)
- class [Child_config](#)

1.5 Input Namespace Reference

Classes

- class [Session](#)

1.6 Nitpicker Namespace Reference

Classes

- class [Session](#)
- class [View](#)

1.7 Pci Namespace Reference

Classes

- class [Device](#)
- class [Session](#)

1.8 Timer Namespace Reference

Classes

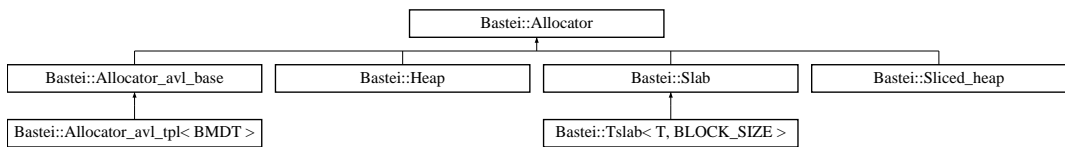
- class [Session](#)

2 Bastei OS Architecture Class Documentation

2.1 Bastei::Allocator Class Reference

```
#include <allocator.h>
```

Inheritance diagram for Bastei::Allocator::



Public Member Functions

- virtual [~Allocator](#) ()
- virtual void * [alloc](#) (size_t size)=0
- virtual void [free](#) (void *addr, size_t size)=0
- virtual size_t [consumed](#) ()
- virtual size_t [overhead](#) ()=0

Classes

- class [Out_of_memory](#)

2.1.1 Constructor & Destructor Documentation

2.1.1.1 virtual Bastei::Allocator::~~Allocator () [inline, virtual]

Destructor

2.1.2 Member Function Documentation

2.1.2.1 virtual void* Bastei::Allocator::alloc (size_t size) [pure virtual]

Allocate block

Implemented in [Bastei::Allocator_avl_base](#), [Bastei::Heap](#), [Bastei::Sliced_heap](#), and [Bastei::Slab](#).

2.1.2.2 virtual void **Bastei::Allocator::free** (void * *addr*, size_t *size*) [pure virtual]

Free block a previously allocated block

Implemented in [Bastei::Allocator_avl_base](#), [Bastei::Heap](#), [Bastei::Sliced_heap](#), and [Bastei::Slab](#).

2.1.2.3 virtual size_t **Bastei::Allocator::consumed** () [inline, virtual]

Return total amount of backingstore consumed by the allocator

Reimplemented in [Bastei::Heap](#), [Bastei::Sliced_heap](#), and [Bastei::Slab](#).

2.1.2.4 virtual size_t **Bastei::Allocator::overhead** () [pure virtual]

Return metadata overhead per block

Implemented in [Bastei::Allocator_avl_base](#), [Bastei::Heap](#), [Bastei::Sliced_heap](#), and [Bastei::Slab](#).

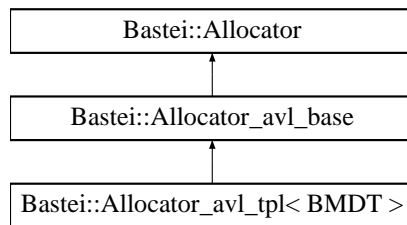
2.2 **Bastei::Allocator::Out_of_memory Class Reference**

```
#include <allocator.h>
```

2.3 **Bastei::Allocator_avl_base Class Reference**

```
#include <allocator_avl.h>
```

Inheritance diagram for `Bastei::Allocator_avl_base`:



Public Member Functions

- int [add_area](#) (addr_t base, size_t size)
- int [remove_area](#) (addr_t base, size_t size)
- void * [alloc_aligned](#) (size_t size, int align=0)
- bool [alloc_addr](#) (size_t size, addr_t addr)
- void [free](#) (void *addr)
- void [dump_addr_tree](#) (Block *addr_node=0)
- bool [any_block_addr](#) (addr_t *out_addr)
- void * [alloc](#) (size_t size)
- void [free](#) (void *addr, size_t)
- size_t [overhead](#) ()
- size_t [avail](#) ()

Protected Member Functions

- [Block * _find_by_address](#) ([addr_t addr](#), [size_t size=0](#), [bool check_overlap=0](#)) [const](#)
- [Allocator_avl_base](#) ([Allocator *md_alloc](#), [size_t md_entry_size](#))

Classes

- class [Block](#)

2.3.1 Constructor & Destructor Documentation

2.3.1.1 [Bastei::Allocator_avl_base::Allocator_avl_base](#) ([Allocator * md_alloc](#), [size_t md_entry_size](#)) [[inline](#), [protected](#)]

Constructor

This constructor can only be called from a derived class that provides an allocator for block metadata entries. This way, we can attach custom information to block metadata.

2.3.2 Member Function Documentation

2.3.2.1 [Block* Bastei::Allocator_avl_base::_find_by_address](#) ([addr_t addr](#), [size_t size = 0](#), [bool check_overlap = 0](#)) [const](#) [[inline](#), [protected](#)]

Find block by specified address

2.3.2.2 [int Bastei::Allocator_avl_base::add_area](#) ([addr_t base](#), [size_t size](#))

Add free address area to allocator

Reimplemented in [Bastei::Allocator_avl_tpl< BMDT >](#).

2.3.2.3 [int Bastei::Allocator_avl_base::remove_area](#) ([addr_t base](#), [size_t size](#))

Remove address area from allocator

2.3.2.4 [void* Bastei::Allocator_avl_base::alloc_aligned](#) ([size_t size](#), [int align = 0](#))

Allocate block

Parameters:

size size of new block

align alignment of new block specified as the power of two.

2.3.2.5 [bool Bastei::Allocator_avl_base::alloc_addr](#) ([size_t size](#), [addr_t addr](#))

Allocate block at address

Parameters:

size size of new block

addr desired address of block

Returns:

True if allocations succeeded, false otherwise

2.3.2.6 void Bastei::Allocator_avl_base::free (void * *addr*)

Free block a previously allocated block

2.3.2.7 void Bastei::Allocator_avl_base::dump_addr_tree (Block * *addr_node* = 0)

Debug hooks

2.3.2.8 bool Bastei::Allocator_avl_base::any_block_addr (addr_t * *out_addr*)

Return address of any block of the allocator

Parameters:

out_addr Result that constains address of block

Returns:

True if block was found or False if there is no block available

If no block was found, *out_addr* is set to zero.

2.3.2.9 void* Bastei::Allocator_avl_base::alloc (size_t *size*) [inline, virtual]

Allocate block

Implements [Bastei::Allocator](#).

2.3.2.10 void Bastei::Allocator_avl_base::free (void * *addr*, size_t *size*) [inline, virtual]

Free block a previously allocated block

Implements [Bastei::Allocator](#).

2.3.2.11 size_t Bastei::Allocator_avl_base::overhead () [inline, virtual]

The overhead is a rough estimation. If a block is somewhere in the middle of a free area, we could consider the metadata for the two free subareas when calculating the overhead.

The 4 represents the overhead of the metadata slab allocator.

Implements [Bastei::Allocator](#).

2.3.2.12 size_t Bastei::Allocator_avl_base::avail ()

Return the sum of available memory

Note that the returned value is not necessarily allocatable because the memory may be fragmented.

2.4 Bastei::Allocator_avl_base::Block Class Reference

```
#include <allocator_avl.h>
```

Public Types

- enum { **FREE** = false, **USED** = true }

Public Member Functions

- bool **higher** (Block *a)
- void **recompute** ()
- int **id** ()
- addr_t **addr** ()
- size_t **avail** ()
- size_t **size** ()
- bool **used** ()
- size_t **max_avail** ()
- void **used** (bool used)
- **Block** ()
- **Block** (addr_t addr, size_t size, bool used)
- **Block** * **find_best_fit** (size_t size, unsigned align=1)
- **Block** * **find_by_address** (addr_t addr, size_t size=0, bool check_overlap=0)
- size_t **avail_in_subtree** (void)
- void **dump** ()
- void **dump_dot** (int indent=0)

2.4.1 Member Enumeration Documentation

2.4.1.1 anonymous enum

Enumerator:

FREE

USED

2.4.2 Constructor & Destructor Documentation

2.4.2.1 Bastei::Allocator_avl_base::Block::Block () [inline]

Constructor

This constructor is called from metadata allocator during initialization of new metadata blocks.

2.4.2.2 Bastei::Allocator_avl_base::Block::Block (addr_t *addr*, size_t *size*, bool *used*) [inline]

Constructor

2.4.3 Member Function Documentation

2.4.3.1 `bool Bastei::Allocator_avl_base::Block::higher (Block * a) [inline]`

Avl_node interface: compare two nodes

2.4.3.2 `void Bastei::Allocator_avl_base::Block::recompute ()`

Avl_node interface: update metadata on node rearrangement

2.4.3.3 `int Bastei::Allocator_avl_base::Block::id () [inline]`

Accessor functions

2.4.3.4 `addr_t Bastei::Allocator_avl_base::Block::addr () [inline]`

2.4.3.5 `size_t Bastei::Allocator_avl_base::Block::avail () [inline]`

2.4.3.6 `size_t Bastei::Allocator_avl_base::Block::size () [inline]`

2.4.3.7 `bool Bastei::Allocator_avl_base::Block::used () [inline]`

2.4.3.8 `size_t Bastei::Allocator_avl_base::Block::max_avail () [inline]`

2.4.3.9 `void Bastei::Allocator_avl_base::Block::used (bool used) [inline]`

2.4.3.10 `Block* Bastei::Allocator_avl_base::Block::find_best_fit (size_t size, unsigned align = 1)`

Find best-fitting block

2.4.3.11 `Block* Bastei::Allocator_avl_base::Block::find_by_address (addr_t addr, size_t size = 0, bool check_overlap = 0)`

Find block that contains the specified address range

2.4.3.12 `size_t Bastei::Allocator_avl_base::Block::avail_in_subtree (void)`

Return sum of available memory in subtree

2.4.3.13 `void Bastei::Allocator_avl_base::Block::dump ()`

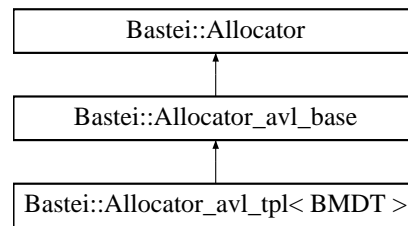
Debug hooks

2.4.3.14 `void Bastei::Allocator_avl_base::Block::dump_dot (int indent = 0)`

2.5 Bastei::Allocator_avl_tpl< BMDT > Class Template Reference

```
#include <allocator_avl.h>
```

Inheritance diagram for Bastei::Allocator_avl_tpl< BMDT >::



Public Member Functions

- [Allocator_avl_tpl](#) ()
- [Allocator_avl_tpl](#) ([Allocator](#) *metadata_chunk_alloc)
- void [metadata](#) (void *addr, BMDT bmd) const
- BMDT * [metadata](#) (void *addr) const
- int [add_area](#) (addr_t base, size_t size)

Classes

- class [Block](#)

2.5.1 Detailed Description

template<typename BMDT> class Bastei::Allocator_avl_tpl< BMDT >

AVL-based allocator with custom metadata attached to each block.

Parameters:

BMDT Block metadata type

2.5.2 Constructor & Destructor Documentation

2.5.2.1 template<typename BMDT> Bastei::Allocator_avl_tpl< BMDT >::Allocator_avl_tpl
() [inline]

Constructor

Use ourself for allocating our metadata blocks. This works only if the managed memory is completely accessible by the allocator. If the allocator is used to manage a non-accessible address range, for example the virtual address space of another task, the metadata allocator can be defined via the second constructor.

2.5.2.2 template<typename BMDT> Bastei::Allocator_avl_tpl< BMDT >::Allocator_avl_tpl
([Allocator](#) * ***metadata_chunk_alloc***) [inline, explicit]

2.5.3 Member Function Documentation

2.5.3.1 `template<typename BMDT> void Bastei::Allocator_avl_tpl< BMDT >::metadata (void * addr, BMDT bmd) const` [inline]

Assign custom metadata to block at specified address

2.5.3.2 `template<typename BMDT> BMDT* Bastei::Allocator_avl_tpl< BMDT >::metadata (void * addr) const` [inline]

Return metadata that was attached to block at specified address

2.5.3.3 `template<typename BMDT> int Bastei::Allocator_avl_tpl< BMDT >::add_area (addr_t base, size_t size)` [inline]

Add free address area to allocator

Reimplemented from [Bastei::Allocator_avl_base](#).

2.6 Bastei::Buffer Class Reference

```
#include <ipc_generic.h>
```

Public Member Functions

- [Buffer](#) (const char **addr*, size_t *size*)
- [Buffer](#) (const char **str*)
- [Buffer](#) ()
- const char * [addr](#) ()
- size_t [size](#) ()

2.6.1 Constructor & Destructor Documentation

2.6.1.1 `Bastei::Buffer::Buffer (const char * addr, size_t size)` [inline]

Construct buffer

2.6.1.2 `Bastei::Buffer::Buffer (const char * str)` [inline, explicit]

Construct buffer from null-terminated string

2.6.1.3 `Bastei::Buffer::Buffer ()` [inline]

Construct an invalid buffer as default

2.6.2 Member Function Documentation

2.6.2.1 `const char* Bastei::Buffer::addr ()` [inline]

2.6.2.2 `size_t Bastei::Buffer::size ()` [inline]

2.7 Bastei::Cap_session Class Reference

```
#include <cap_session.h>
```

Public Member Functions

- virtual `~Cap_session ()`
- virtual `Capability alloc (Capability ep_cap)=0`
- virtual `void free (Capability cap)=0`

Protected Types

- enum `Opcode { ALLOC, FREE }`

2.7.1 Member Enumeration Documentation

2.7.1.1 enum `Bastei::Cap_session::Opcode` [protected]

Enumerator:

`ALLOC`
`FREE`

2.7.2 Constructor & Destructor Documentation

2.7.2.1 virtual `Bastei::Cap_session::~~Cap_session ()` [inline, virtual]

2.7.3 Member Function Documentation

2.7.3.1 virtual `Capability Bastei::Cap_session::alloc (Capability ep_cap)` [pure virtual]

Allocate new unique userland capability

Parameters:

`ep_cap` entrypoint that will use this capability

Returns:

new userland capability

2.7.3.2 virtual `void Bastei::Cap_session::free (Capability cap)` [pure virtual]

Free userland capability

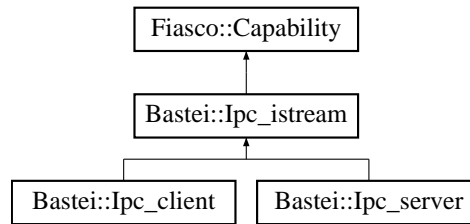
Parameters:

`cap` userland capability to free

2.8 Fiasco::Capability Class Reference

```
#include <capability.h>
```

Inheritance diagram for Fiasco::Capability::



Public Member Functions

- [Capability](#) ()
- long [local_name](#) () const
- bool [valid](#) () const
- [Capability](#) (l4_threadid_t tid, l4_umword_t local_name)
- l4_threadid_t [tid](#) () const

Protected Attributes

- l4_threadid_t [_tid](#)
- long [_local_name](#)

2.8.1 Constructor & Destructor Documentation

2.8.1.1 Fiasco::Capability::Capability () [inline]

Default constructor

2.8.1.2 Fiasco::Capability::Capability (l4_threadid_t tid, l4_umword_t local_name) [inline]

Constructor

This constructor can be called to create a [Fiasco](#) capability by hand. It must never be used from generic code!

2.8.2 Member Function Documentation

2.8.2.1 long Fiasco::Capability::local_name () const [inline]

2.8.2.2 bool Fiasco::Capability::valid () const [inline]

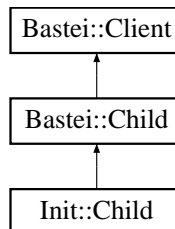
2.8.2.3 `l4_threadid_t Fiasco::Capability::tid () const` [inline]

Access raw capability data

2.8.3 Member Data Documentation**2.8.3.1 `l4_threadid_t Fiasco::Capability::_tid` [protected]****2.8.3.2 `long Fiasco::Capability::_local_name` [protected]****2.9 Bastei::Child Class Reference**

```
#include <child.h>
```

Inheritance diagram for Bastei::Child:

**Public Member Functions**

- [Child](#) (const char *name, [Capability](#) elf_ds_cap, [Capability](#) ram_session_cap, [Capability](#) cpu_session_cap, [Cap_session](#) *cap_session, char *args[])
- [~Child](#) ()
- [Allocator](#) * [heap](#) ()
- [Capability](#) [ram_session_cap](#) ()
- [Capability](#) [cpu_session_cap](#) ()
- void [finalize_construction](#) ()
- void [prepare_destruction](#) ()
- void [revoke_server](#) ([Child](#) *server)
- [Server_entrypoint](#) * [parent_entrypoint](#) ()
- int [announce](#) (const char *service_name, [Capability](#) service_root)
- [Capability](#) [session](#) (const char *service_name, const char *args)
- int [transfer_quota](#) ([Capability](#) to_session_cap, size_t amount)
- void [close](#) ([Capability](#) session_cap)
- void [exit](#) (int exit_value)

Protected Member Functions

- int [_add_session](#) ([Session](#) s)
- void [_remove_session](#) ([Session](#) *s)

Protected Attributes

- [Capability _ram_session_cap](#)
- [Ram_session_client _ram_session_client](#)

2.9.1 Constructor & Destructor Documentation

2.9.1.1 Bastei::Child::Child (`const char * name`, `Capability elf_ds_cap`, `Capability ram_session_cap`, `Capability cpu_session_cap`, `Cap_session * cap_session`, `char * args[]`)
[inline]

Constructor

Parameters:

- name* Unique name of the child
- elf_ds_cap* [Dataspaces](#) containing the binary
- ram_session_cap* RAM session with the child's quota
- cpu_session_cap* CPU session with the child's quota
- cap_session* Capability allocator
- args* ???

2.9.1.2 Bastei::Child::~~Child () [inline]

Destructor

On destruction of a child, we close all sessions of the child to other services.

Reimplemented in [Init::Child](#).

2.9.2 Member Function Documentation

2.9.2.1 int Bastei::Child::_add_session (`Session s`) [inline, protected]

2.9.2.2 void Bastei::Child::_remove_session (`Session * s`) [inline, protected]

2.9.2.3 Allocator* Bastei::Child::heap () [inline]

Return heap that uses the child's quota

2.9.2.4 Capability Bastei::Child::ram_session_cap () [inline]

2.9.2.5 Capability Bastei::Child::cpu_session_cap () [inline]

2.9.2.6 void Bastei::Child::finalize_construction () [inline]

Finalize construction of child

On construction of the child, we create a new task and a new parent server thread. The new child tasks starts immediately to speak with its parent server thread and performs [session\(\)](#) RPC calls. Therefore, the dispatch function of the parent server may be called before the complete child object (including its final vtable entries) is constructed. Therefore, we delay the server activation of the parent server until all constructors are completed.

2.9.2.7 void Bastei::Child::prepare_destruction () [inline]

Prepare destruction of child

Similar to the construction case we must ensure that the server activation will not use half-destructed objects (vtable). Therefore we dissolve the server object (this child) from entrypoint before destruction starts.

2.9.2.8 void Bastei::Child::revoke_server (Child * server) [inline]

Discard all sessions to specified server

When this function is called, we assume the server task to be dead and all that all server quote was already transfered back to our own [env\(\)->ram_session\(\)](#) account. Note that the specified server object may not exist anymore. We do not de-reference the server argument in here!

2.9.2.9 Server_entrypoint* Bastei::Child::parent_entrypoint () [inline]

Request entrypoint used for the child's parent interface

This function allows for attaching additional server objects to the child-specific entrypoint, for example to provide child-specific services without the need to create separate entrypoints and activations. Note that all calls to this entrypoint get serialized. Therefore, a blocking server function defers calls to the parent interface.

2.9.2.10 int Bastei::Child::announce (const char * service_name, Capability service_root)
[inline]

Reimplemented in [Init::Child](#).

2.9.2.11 Capability Bastei::Child::session (const char * service_name, const char * args)
[inline]

By default, we only check for the sessions that we created for the child on startup. A derived class may implement a proper service resolving policy.

Reimplemented in [Init::Child](#).

2.9.2.12 int Bastei::Child::transfer_quota (Capability to_session_cap, size_t amount)
[inline]**2.9.2.13 void Bastei::Child::close (Capability session_cap)** [inline]

2.9.2.14 void `Bastei::Child::exit (int exit_value)` [`inline`]

2.9.3 Member Data Documentation

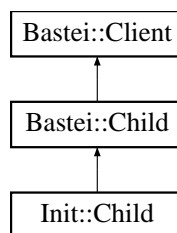
2.9.3.1 Capability `Bastei::Child::_ram_session_cap` [`protected`]

2.9.3.2 `Ram_session_client` `Bastei::Child::_ram_session_client` [`protected`]

2.10 `Init::Child` Class Reference

```
#include <child.h>
```

Inheritance diagram for `Init::Child`:



Public Member Functions

- `Child` (`const char *name`, `Bastei::Capability elf_ds_cap`, `Bastei::Capability ram_session_cap`, `Bastei::Capability cpu_session_cap`, `Bastei::Cap_session *cap_session`, `Bastei::Remote_service_pool *remote_services`, `Bastei::Capability config_ds_cap`)
- `~Child` ()
- `const char * name` ()
- `Bastei::Capability session` (`const char *service_name`, `const char *args`)
- `int announce` (`const char *service_name`, `Bastei::Capability service_root`)

Protected Member Functions

- `void _prepend_label` (`const char *args`)
- `virtual bool _delegate_to_parent` (`const char *service_name`, `const char *args`)

Static Protected Member Functions

- `static Bastei::size_t _ram_quota` (`const char *args`)

Protected Attributes

- `Bastei::Remote_service_pool * _remote_services`
- `char _argbuf` [256]

Classes

- class `Config_rom_session_component`

2.10.1 Constructor & Destructor Documentation

2.10.1.1 `Init::Child::Child (const char * name, Bastei::Capability elf_ds_cap, Bastei::Capability ram_session_cap, Bastei::Capability cpu_session_cap, Bastei::Cap_session * cap_session, Bastei::Remote_service_pool * remote_services, Bastei::Capability config_ds_cap) [inline]`

Constructor

Parameters:

name Name of child (used for debugging)

elf_ds_cap ELF binary of program to run

ram_session_cap RAM session for child

cpu_session_cap CPU session for child

cap_session Capability session for child

remote_services Service pool for child

config_ds_cap Capability for configuration dataspace

See the description of [Bastei::Child](#) for the other arguments.

2.10.1.2 `Init::Child::~~Child () [inline]`

Destructor

Reimplemented from [Bastei::Child](#).

2.10.2 Member Function Documentation

2.10.2.1 `static Bastei::size_t Init::Child::_ram_quota (const char * args) [inline, static, protected]`

2.10.2.2 `void Init::Child::_prepend_label (const char * args) [inline, protected]`

2.10.2.3 `virtual bool Init::Child::_delegate_to_parent (const char * service_name, const char * args) [inline, protected, virtual]`

Decide if a session request should be delegated to our parent

We should do this only if we are confident that such a service exists at our parent. If the a service is not known to our parent, the parent may enqueue us into his wait-for-service queue and block the request.

2.10.2.4 `const char* Init::Child::name ()` [inline, virtual]

Return name of client

This function is here for debugging purposes.

Reimplemented from [Bastei::Client](#).

2.10.2.5 `Bastei::Capability Init::Child::session (const char * service_name, const char * args)` [inline]

By default, we only check for the sessions that we created for the child on startup. A derived class may implement a proper service resolving policy.

Reimplemented from [Bastei::Child](#).

2.10.2.6 `int Init::Child::announce (const char * service_name, Bastei::Capability service_root)` [inline]

Reimplemented from [Bastei::Child](#).

2.10.3 Member Data Documentation

2.10.3.1 `Bastei::Remote_service_pool* Init::Child::_remote_services` [protected]

2.10.3.2 `char Init::Child::_argbuf[256]` [protected]

2.11 *Init::Child_config* Class Reference

```
#include <child_config.h>
```

Public Member Functions

- [Child_config](#) ([Bastei::Capability](#) ram_session_cap, [Bastei::Xml_node](#) start_node)
- [~Child_config](#) ()
- [Bastei::Capability](#) dataspace ()

2.11.1 Constructor & Destructor Documentation

2.11.1.1 `Init::Child_config::Child_config (Bastei::Capability ram_session_cap, Bastei::Xml_node start_node)` [inline]

Constructor

The provided RAM session is used to obtain a dataspace for holding the copy of the child's configuration data. Normally, the child's RAM session should be used to account the consumed RAM quota to the child.

2.11.1.2 Init::Child_config::~~Child_config () [inline]

Destructor

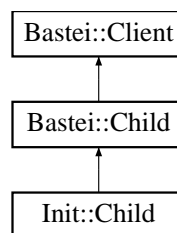
2.11.2 Member Function Documentation**2.11.2.1 Bastei::Capability Init::Child_config::dataspace ()** [inline]

Request dataspace holding the start node's configuration data

2.12 Bastei::Client Class Reference

#include <service.h>

Inheritance diagram for Bastei::Client:

**Public Member Functions**

- [Client \(\)](#)
- virtual [~Client \(\)](#)
- virtual const char * [name \(\)](#)
- void [apply_for](#) (const char *apply_for)
- const char * [apply_for \(\)](#)
- void [sleep \(\)](#)
- void [wakeup \(\)](#)

2.12.1 Detailed Description

A client is someone who applies for a service. If the service is not available yet, we enqueue the client into a wait queue and wake him up as soon as the requested service gets available.

2.12.2 Constructor & Destructor Documentation**2.12.2.1 Bastei::Client::Client ()** [inline]

Constructor

2.12.2.2 virtual Bastei::Client::~~Client () [inline, virtual]

2.12.3 Member Function Documentation

2.12.3.1 virtual const char* **Bastei::Client::name** () [inline, virtual]

Return name of client

This function is here for debugging purposes.

Reimplemented in [Init::Child](#).

2.12.3.2 void **Bastei::Client::apply_for** (const char * *apply_for*) [inline]

Set/Request service name that we are currently applying for

2.12.3.3 const char* **Bastei::Client::apply_for** () [inline]

2.12.3.4 void **Bastei::Client::sleep** () [inline]

[Service](#) wait queue support

2.12.3.5 void **Bastei::Client::wakeup** () [inline]

2.13 **Bastei::Cpu_session Class Reference**

```
#include <cpu_session.h>
```

Public Types

- enum { [THREAD_NAME_LEN](#) = 32 }

Public Member Functions

- virtual [~Cpu_session](#) ()
- virtual int [create_thread](#) (const char *name, [Capability](#) *new_thread_cap)=0
- virtual void [kill_thread](#) ([Capability](#) thread_cap)=0
- virtual [Capability](#) [first](#) ()=0
- virtual [Capability](#) [next](#) ([Capability](#) curr)=0
- virtual int [set_pager](#) ([Capability](#) thread_cap, [Capability](#) pager_cap)=0
- virtual int [start](#) ([Capability](#) thread_cap, addr_t ip, addr_t sp)=0
- virtual int [cancel_blocking](#) ([Capability](#) thread_cap)=0
- virtual int [name](#) ([Capability](#) thread_cap, char *name_dst, size_t name_len)=0
- virtual int [state](#) ([Capability](#) thread_cap, [Thread_state](#) *state_dst)=0

Protected Types

- enum `Opcode` {
 `CREATE_THREAD`, `KILL_THREAD`, `FIRST`, `NEXT`,
 `SET_PAGER`, `START`, `CANCEL_BLOCKING`, `NAME`,
 `STATE` }

2.13.1 Member Enumeration Documentation

2.13.1.1 enum `Bastei::Cpu_session::Opcode` [protected]

Enumerator:

`CREATE_THREAD`
`KILL_THREAD`
`FIRST`
`NEXT`
`SET_PAGER`
`START`
`CANCEL_BLOCKING`
`NAME`
`STATE`

2.13.1.2 anonymous enum

Enumerator:

`THREAD_NAME_LEN`

2.13.2 Constructor & Destructor Documentation

2.13.2.1 virtual `Bastei::Cpu_session::~~Cpu_session ()` [inline, virtual]

2.13.3 Member Function Documentation

2.13.3.1 virtual `int Bastei::Cpu_session::create_thread (const char * name, Capability * new_thread_cap)` [pure virtual]

Create a new thread

Parameters:

name name for the thread

Return values:

new_thread_cap capability for the new thread

2.13.3.2 virtual void Bastei::Cpu_session::kill_thread (Capability *thread_cap*) [pure virtual]

Kill an existing thread

Parameters:

thread_cap capability of the thread to kill

2.13.3.3 virtual Capability Bastei::Cpu_session::first () [pure virtual]

Retrieve thread list of CPU session

The [next\(\)](#) function returns an invalid capability if the specified thread does not exist or if it is the last one of the CPU session.

2.13.3.4 virtual Capability Bastei::Cpu_session::next (Capability *curr*) [pure virtual]

2.13.3.5 virtual int Bastei::Cpu_session::set_pager (Capability *thread_cap*, Capability *pager_cap*) [pure virtual]

Set paging capabilities for thread

Parameters:

thread_cap [Thread](#) to configure

pager_cap Capability used to propagate page faults

2.13.3.6 virtual int Bastei::Cpu_session::start (Capability *thread_cap*, addr_t *ip*, addr_t *sp*) [pure virtual]

Modify instruction and stack pointer of thread - start the thread

Parameters:

thread_cap [Thread](#) to start

ip Initial instruction pointer

sp Initial stack pointer

Returns:

0 on success

2.13.3.7 virtual int Bastei::Cpu_session::cancel_blocking (Capability *thread_cap*) [pure virtual]

Cancel a currently blocking operation

Parameters:

thread_cap [Thread](#) to unblock

Returns:

0 on success

2.13.3.8 virtual int Bastei::Cpu_session::name (Capability *thread_cap*, char * *name_dst*, size_t *name_len*) [pure virtual]

Return thread name

Parameters:

- thread_cap* [Thread](#) to query
- name_dst* Destination string buffer
- name_len* Length of destination string buffer

Returns:

0 on success

2.13.3.9 virtual int Bastei::Cpu_session::state (Capability *thread_cap*, Thread_state * *state_dst*) [pure virtual]

Return thread state

Parameters:

- thread_cap* [Thread](#) to spy on
- state_dst* Result

Returns:

0 on success

2.14 Bastei::Dataspaces Class Reference

```
#include <dataspaces.h>
```

Public Types

- enum [Opcode](#) { [SIZE](#), [PHYS_ADDR](#), [NUM_GENERIC_OPCODES](#) }

Public Member Functions

- virtual [~Dataspaces](#) ()
- virtual size_t [size](#) ()=0
- virtual addr_t [phys_addr](#) ()=0

2.14.1 Member Enumeration Documentation

2.14.1.1 enum Bastei::Dataspaces::Opcode

Enumerator:

SIZE
PHYS_ADDR
NUM_GENERIC_OPCODES

2.14.2 Constructor & Destructor Documentation

2.14.2.1 virtual `Bastei::Dataspace::~~Dataspace ()` [inline, virtual]

2.14.3 Member Function Documentation

2.14.3.1 virtual `size_t Bastei::Dataspace::size ()` [pure virtual]

Request size of dataspace

2.14.3.2 virtual `addr_t Bastei::Dataspace::phys_addr ()` [pure virtual]

Request base address in physical address space

2.15 `Pci::Device` Class Reference

```
#include <pci_device.h>
```

Public Types

- enum { `NUM_RESOURCES` = 6 }

Public Member Functions

- virtual `~Device ()`
- virtual unsigned short `device_id ()`=0
- virtual unsigned `class_code ()`=0
- virtual `Resource resource (int resource_id)`=0
- unsigned `base_class ()`
- unsigned `sub_class ()`

Protected Types

- enum `Opcode` { `DEVICE_ID`, `CLASS_CODE`, `RESOURCE` }

Classes

- class `Resource`

2.15.1 Member Enumeration Documentation

2.15.1.1 enum `Pci::Device::Opcode` [protected]

Enumerator:

`DEVICE_ID`
`CLASS_CODE`

RESOURCE

2.15.1.2 anonymous enum

[Resource](#) type, either port I/O resource or memory-mapped resource

Enumerator:

NUM_RESOURCES

2.15.2 Constructor & Destructor Documentation

2.15.2.1 virtual `Pci::Device::~~Device ()` [inline, virtual]

2.15.3 Member Function Documentation

2.15.3.1 virtual unsigned short `Pci::Device::device_id ()` [pure virtual]

Return device ID obtained from the PCI config space

2.15.3.2 virtual unsigned `Pci::Device::class_code ()` [pure virtual]

Return device class code from the PCI config space

2.15.3.3 virtual Resource `Pci::Device::resource (int resource_id)` [pure virtual]

Query PCI-resource information

Parameters:

resource_id Index of according PCI resource of the device

Returns:

[Resource](#) description. If the supplied resource ID is invalid, the type of the returned resource is INVALID.

2.15.3.4 unsigned `Pci::Device::base_class ()` [inline]

2.15.3.5 unsigned `Pci::Device::sub_class ()` [inline]

2.16 `Pci::Device::Resource` Class Reference

```
#include <pci_device.h>
```

Public Types

- enum [Type](#) { [IO](#), [MEMORY](#), [INVALID](#) }

Public Member Functions

- [Resource](#) ()
- [Resource](#) (unsigned bar, unsigned size)
- unsigned [base](#) ()
- unsigned [size](#) ()
- bool [prefetchable](#) ()
- [Type](#) [type](#) ()

2.16.1 Member Enumeration Documentation

2.16.1.1 enum `Pci::Device::Resource::Type`

Enumerator:

IO
MEMORY
INVALID

2.16.2 Constructor & Destructor Documentation

2.16.2.1 `Pci::Device::Resource::Resource ()` [inline]

Default constructor

2.16.2.2 `Pci::Device::Resource::Resource (unsigned bar, unsigned size)` [inline]

Constructor

Parameters:

bar Content of base-address register
size [Resource](#) size

This constructor is only used by the PCI bus driver that implements the server-side of the the PCI session. If bar is set to zero, the constucted resource description represents an INVALID resource.

2.16.3 Member Function Documentation

2.16.3.1 `unsigned Pci::Device::Resource::base ()` [inline]

Return base address of resource

2.16.3.2 `unsigned Pci::Device::Resource::size ()` [inline]

Return resource size in bytes

2.16.3.3 `bool Pci::Device::Resource::prefetchable ()` [inline]

Return true if resource is prefetchable memory

2.16.3.4 Type `Pci::Device::Resource::type ()` [inline]

Return resource type

2.17 Bastei::Empty Class Reference

```
#include <allocator_avl.h>
```

2.17.1 Detailed Description

Define AVL-based allocator without any metadata attached to each block.

2.18 Bastei::Env Class Reference

```
#include <env.h>
```

Public Member Functions

- virtual `~Env ()`
- virtual `Parent * parent ()=0`
- virtual `Ram_session * ram_session ()=0`
- virtual `Capability ram_session_cap ()=0`
- virtual `Cpu_session * cpu_session ()=0`
- virtual `Rm_session * rm_session ()=0`
- virtual `Task_session * task_session ()=0`
- virtual `Allocator * heap ()=0`

2.18.1 Constructor & Destructor Documentation

2.18.1.1 virtual `Bastei::Env::~~Env ()` [inline, virtual]

2.18.2 Member Function Documentation

2.18.2.1 virtual `Parent* Bastei::Env::parent ()` [pure virtual]

Communication channel to our parent

2.18.2.2 virtual `Ram_session* Bastei::Env::ram_session ()` [pure virtual]

RAM session for the program

The RAM [Session](#) represents a quota of memory that is available to the program. Quota can be used to allocate RAM-Dataspaces.

2.18.2.3 virtual `Capability Bastei::Env::ram_session_cap ()` [pure virtual]

2.18.2.4 virtual Cpu_session* Bastei::Env::cpu_session () [pure virtual]

CPU session for the program

This session is used to create threads.

2.18.2.5 virtual Rm_session* Bastei::Env::rm_session () [pure virtual]

Region manager session of the program

2.18.2.6 virtual Task_session* Bastei::Env::task_session () [pure virtual]

[Task](#) session of the program

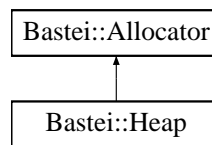
2.18.2.7 virtual Allocator* Bastei::Env::heap () [pure virtual]

[Heap](#) backed by the ram_session of the environment.

2.19 Bastei::Heap Class Reference

```
#include <heap.h>
```

Inheritance diagram for Bastei::Heap::

**Public Types**

- enum { [UNLIMITED](#) = ~0 }

Public Member Functions

- [Heap](#) ([Ram_session](#) *ram_session, [Rm_session](#) *rm_session, size_t quota_limit=UNLIMITED, void *static_addr=0, size_t static_size=0)
- int [quota_limit](#) (size_t new_quota_limit)
- void * [alloc](#) (size_t size)
- void [free](#) (void *addr, size_t size)
- size_t [consumed](#) ()
- size_t [overhead](#) ()

Classes

- class [Dataspace](#)
- class [Dataspace_pool](#)

2.19.1 Member Enumeration Documentation

2.19.1.1 anonymous enum

Enumerator:

UNLIMITED

2.19.2 Constructor & Destructor Documentation

2.19.2.1 Bastei::Heap::Heap (Ram_session * *ram_session*, Rm_session * *rm_session*, size_t *quota_limit* = UNLIMITED, void * *static_addr* = 0, size_t *static_size* = 0) [inline]

2.19.3 Member Function Documentation

2.19.3.1 int Bastei::Heap::quota_limit (size_t *new_quota_limit*)

Reconfigure quota limit

Returns:

negative error code if new quota limit is higher than currently used quota.

2.19.3.2 void* Bastei::Heap::alloc (size_t *size*) [virtual]

Allocate block

Implements [Bastei::Allocator](#).

2.19.3.3 void Bastei::Heap::free (void * *addr*, size_t *size*) [virtual]

Free block a previously allocated block

Implements [Bastei::Allocator](#).

2.19.3.4 size_t Bastei::Heap::consumed () [inline, virtual]

Return total amount of backingstore consumed by the allocator

Reimplemented from [Bastei::Allocator](#).

2.19.3.5 size_t Bastei::Heap::overhead () [inline, virtual]

Return metadata overhead per block

Implements [Bastei::Allocator](#).

2.20 Bastei::io_mem_session Class Reference

```
#include <io_mem_session.h>
```

Public Member Functions

- virtual `~Io_mem_session ()`
- virtual `Capability dataspace ()=0`

Protected Types

- enum `Opcode { DATASPACE }`

2.20.1 Member Enumeration Documentation

2.20.1.1 enum `Bastei::Io_mem_session::Opcode` [protected]

Enumerator:

`DATASPACE`

2.20.2 Constructor & Destructor Documentation

2.20.2.1 virtual `Bastei::Io_mem_session::~Io_mem_session ()` [inline, virtual]

2.20.3 Member Function Documentation

2.20.3.1 virtual `Capability Bastei::Io_mem_session::dataspace ()` [pure virtual]

Request dataspace containing the IO_MEM session data

Returns:

Capability to IO_MEM dataspace The capability may be invalid.

2.21 *Bastei::Io_port_session Class Reference*

```
#include <io_port_session.h>
```

Public Member Functions

- virtual `~Io_port_session ()`
- virtual unsigned char `inb (unsigned short address)=0`
- virtual unsigned short `inw (unsigned short address)=0`
- virtual unsigned `inl (unsigned short address)=0`
- virtual void `outb (unsigned short address, unsigned char value)=0`
- virtual void `outw (unsigned short address, unsigned short value)=0`
- virtual void `outl (unsigned short address, unsigned value)=0`

Protected Types

- enum `Opcode` {
 `INB`, `INW`, `INL`, `OUTB`,
 `OUTW`, `OUTL` }

2.21.1 Member Enumeration Documentation

2.21.1.1 enum `Bastei::Io_port_session::Opcode` [protected]

Enumerator:

INB

INW

INL

OUTB

OUTW

OUTL

2.21.2 Constructor & Destructor Documentation

2.21.2.1 virtual `Bastei::Io_port_session::~~Io_port_session ()` [inline, virtual]

2.21.3 Member Function Documentation

2.21.3.1 virtual unsigned char `Bastei::Io_port_session::inb (unsigned short address)` [pure virtual]

Read byte (8 bit)

Parameters:

address Physical I/O port address

Returns:

Value read from port

2.21.3.2 virtual unsigned short `Bastei::Io_port_session::inw (unsigned short address)` [pure virtual]

Read word (16 bit)

Parameters:

address Physical I/O port address

Returns:

Value read from port

2.21.3.3 virtual unsigned Bastei::lo_port_session::inl (unsigned short *address*) [pure virtual]

Read double word (32 bit)

Parameters:

address Physical I/O port address

Returns:

Value read from port

2.21.3.4 virtual void Bastei::lo_port_session::outb (unsigned short *address*, unsigned char *value*) [pure virtual]

Write byte (8 bit)

Parameters:

address Physical I/O port address

value Value to write to port

2.21.3.5 virtual void Bastei::lo_port_session::outw (unsigned short *address*, unsigned short *value*) [pure virtual]

Write word (16 bit)

Parameters:

address Physical I/O port address

value Value to write to port

2.21.3.6 virtual void Bastei::lo_port_session::outl (unsigned short *address*, unsigned *value*) [pure virtual]

Write double word (32 bit)

Parameters:

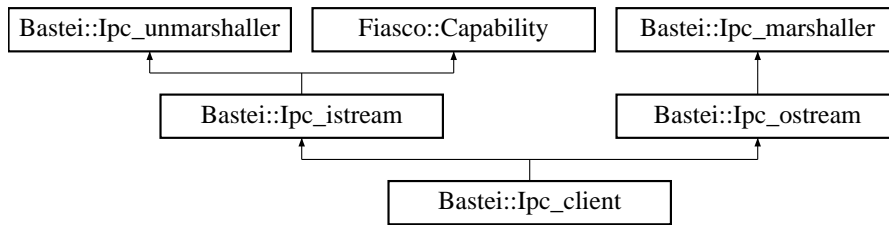
address Physical I/O port address

value Value to write to port

2.22 *Bastei::ipc_client* Class Reference

```
#include <ipc_generic.h>
```

Inheritance diagram for *Bastei::Ipc_client*:



Public Member Functions

- `Ipc_client` (`Capability &srv`, `Msgbuf_base *snd_msg`, `Msgbuf_base *rcv_msg`)
- `Ipc_client & operator<<` (`Ipc_client_call`)
- `template<typename T>`
`Ipc_client & operator<<` (`T value`)
- `Ipc_client & operator<<` (`Buffer b`)
- `template<typename T>`
`Ipc_client & operator>>` (`T &value`)
- `Ipc_client & operator>>` (`Buffer &b`)
- `operator int` ()

Protected Member Functions

- `void _prepare_next_call` ()
- `void _call` ()

Protected Attributes

- `int _result`

2.22.1 Constructor & Destructor Documentation

2.22.1.1 `Bastei::Ipc_client::Ipc_client` (`Capability & srv`, `Msgbuf_base * snd_msg`, `Msgbuf_base * rcv_msg`)

Constructor

2.22.2 Member Function Documentation

2.22.2.1 `void Bastei::Ipc_client::_prepare_next_call` () [protected]

2.22.2.2 `void Bastei::Ipc_client::_call` () [protected]

Send RPC message and wait for result

2.22.2.3 `Ipc_client& Bastei::Ipc_client::operator<<` (`Ipc_client_call`) [inline]

Operator that issues an IPC call

2.22.2.4 `template<typename T> Ipc_client& Bastei::lpc_client::operator<< (T value)`
 [inline]

Insert value into send buffer

Reimplemented from [Bastei::Ipc_ostream](#).

2.22.2.5 `Ipc_client& Bastei::lpc_client::operator<< (Buffer b)` [inline]

Insert byte buffer to send buffer

Reimplemented from [Bastei::Ipc_ostream](#).

2.22.2.6 `template<typename T> Ipc_client& Bastei::lpc_client::operator>> (T & value)`
 [inline]

Read values from receive buffer

Reimplemented from [Bastei::Ipc_istream](#).

2.22.2.7 `Ipc_client& Bastei::lpc_client::operator>> (Buffer & b)` [inline]

Read byte buffer from receive buffer

Reimplemented from [Bastei::Ipc_istream](#).

2.22.2.8 `Bastei::lpc_client::operator int ()` [inline]

2.22.3 Member Data Documentation

2.22.3.1 `int Bastei::Ipc_client::_result` [protected]

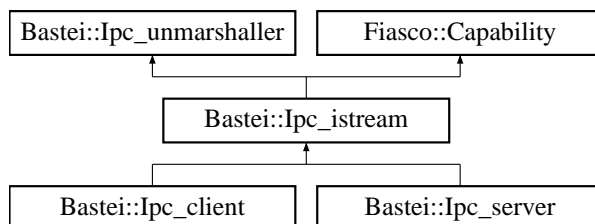
2.23 Bastei::lpc_error Class Reference

```
#include <ipc_generic.h>
```

2.24 Bastei::lpc_istream Class Reference

```
#include <ipc_generic.h>
```

Inheritance diagram for Bastei::Ipc_istream::



Public Member Functions

- [Ipc_istream \(Msgbuf_base *rcv_msg\)](#)
- [~Ipc_istream \(\)](#)
- [long badge \(\)](#)
- [Ipc_istream & operator>> \(Ipc_istream_wait\)](#)
- [template<typename T> Ipc_istream & operator>> \(T &value\)](#)
- [Ipc_istream & operator>> \(Buffer &b\)](#)

Protected Member Functions

- [void _prepare_next_receive \(\)](#)
- [void _wait \(\)](#)

Protected Attributes

- [Msgbuf_base * _rcv_msg](#)
- [Connection_state_rcv_cs](#)

2.24.1 Detailed Description

Stream for receiving information

2.24.2 Constructor & Destructor Documentation

2.24.2.1 [Bastei::lpc_istream::lpc_istream \(Msgbuf_base * *rcv_msg*\)](#) [explicit]

2.24.2.2 [Bastei::lpc_istream::~~lpc_istream \(\)](#)

2.24.3 Member Function Documentation

2.24.3.1 [void Bastei::lpc_istream::_prepare_next_receive \(\)](#) [protected]

Reset unmarshaller

2.24.3.2 [void Bastei::lpc_istream::_wait \(\)](#) [protected]

Wait for incoming message to be received in `_rcv_msg`

2.24.3.3 [long Bastei::lpc_istream::badge \(\)](#) [inline]

Read badge that was supplied with the message

2.24.3.4 [Ipc_istream& Bastei::lpc_istream::operator>> \(Ipc_istream_wait\)](#) [inline]

Block for an incoming message filling the receive buffer

2.24.3.5 `template<typename T> Ipc_istream& Bastei::Ipc_istream::operator>> (T & value)`
 [inline]

Read values from receive buffer

Reimplemented in [Bastei::Ipc_client](#), and [Bastei::Ipc_server](#).

2.24.3.6 `Ipc_istream& Bastei::Ipc_istream::operator>> (Buffer & b)` [inline]

Read byte buffer from receive buffer

Reimplemented in [Bastei::Ipc_client](#).

2.24.4 Member Data Documentation

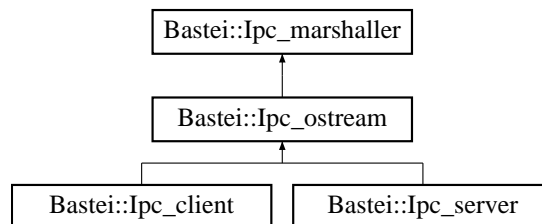
2.24.4.1 `Msgbuf_base* Bastei::Ipc_istream::_rcv_msg` [protected]

2.24.4.2 `Connection_state Bastei::Ipc_istream::_rcv_cs` [protected]

2.25 Bastei::Ipc_marshall Class Reference

`#include <ipc_generic.h>`

Inheritance diagram for `Bastei::Ipc_marshall`:



Public Member Functions

- [Ipc_marshall](#) (char *sndbuf, size_t sndbuf_size)

Protected Member Functions

- `template<typename T>`
 void [_write_to_buf](#) (T &value)
- void [_write_to_buf](#) (const char *src_addr, unsigned num_bytes)
- void [_write_buffer_to_buf](#) (Buffer b)

Protected Attributes

- char * [_sndbuf](#)
- size_t [_sndbuf_size](#)

- unsigned `_write_offset`

2.25.1 Detailed Description

Marshal arguments into send message buffer

2.25.2 Constructor & Destructor Documentation

2.25.2.1 `Bastei::lpc_marshall::lpc_marshall (char * sndbuf, size_t sndbuf_size)`
[inline]

2.25.3 Member Function Documentation

2.25.3.1 `template<typename T> void Bastei::lpc_marshall::_write_to_buf (T & value)`
[inline, protected]

Write value to send buffer

2.25.3.2 `void Bastei::lpc_marshall::_write_to_buf (const char * src_addr, unsigned num_bytes)` [inline, protected]

Write bytes to send buffer

2.25.3.3 `void Bastei::lpc_marshall::_write_buffer_to_buf (Buffer b)` [inline, protected]

Write `Buffer` to send buffer

2.25.4 Member Data Documentation

2.25.4.1 `char* Bastei::lpc_marshall::_sndbuf` [protected]

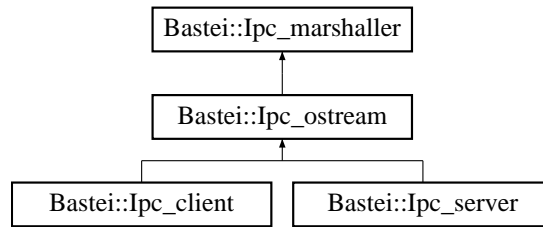
2.25.4.2 `size_t Bastei::lpc_marshall::_sndbuf_size` [protected]

2.25.4.3 `unsigned Bastei::lpc_marshall::_write_offset` [protected]

2.26 `Bastei::lpc_ostream` Class Reference

```
#include <ipc_generic.h>
```

Inheritance diagram for `Bastei::lpc_ostream`:



Public Member Functions

- [Ipc_ostream](#) ([Capability](#) dst, [Msgbuf_base](#) *snd_msg)
- [bool ready_for_send](#) ()
- [template<typename T>](#)
[Ipc_ostream](#) & [operator<<](#) (T value)
- [Ipc_ostream](#) & [operator<<](#) ([Buffer](#) b)
- [Ipc_ostream](#) & [operator<<](#) ([Ipc_ostream_send](#))

Protected Member Functions

- [void _prepare_next_send](#) ()
- [void _send](#) ()

Protected Attributes

- [Msgbuf_base](#) * [_snd_msg](#)
- [Capability](#) [_dst](#)

2.26.1 Detailed Description

Stream for sending information via a capability to an endpoint

2.26.2 Constructor & Destructor Documentation

2.26.2.1 [Bastei::Ipc_ostream::Ipc_ostream](#) ([Capability](#) *dst*, [Msgbuf_base](#) * *snd_msg*)

Constructor

2.26.3 Member Function Documentation

2.26.3.1 [void Bastei::Ipc_ostream::_prepare_next_send](#) () [protected]

Reset marshaller and write badge at the beginning of the message

2.26.3.2 [void Bastei::Ipc_ostream::_send](#) () [protected]

Send message in [_snd_msg](#) to [_dst](#)

2.26.3.3 bool Bastei::Ipc_ostream::ready_for_send () [inline]

Return true if [Ipc_ostream](#) is ready for send

2.26.3.4 template<typename T> Ipc_ostream& Bastei::Ipc_ostream::operator<< (T value) [inline]

Insert value into send buffer

Reimplemented in [Bastei::Ipc_client](#), and [Bastei::Ipc_server](#).

2.26.3.5 Ipc_ostream& Bastei::Ipc_ostream::operator<< (Buffer b) [inline]

Insert byte buffer to send buffer

Reimplemented in [Bastei::Ipc_client](#).

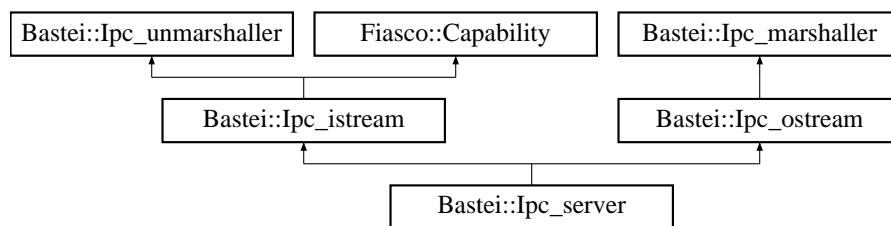
2.26.3.6 Ipc_ostream& Bastei::Ipc_ostream::operator<< (Ipc_ostream_send) [inline]

Issue the sending of the message buffer

2.26.4 Member Data Documentation**2.26.4.1 Msgbuf_base* Bastei::Ipc_ostream::_snd_msg** [protected]**2.26.4.2 Capability Bastei::Ipc_ostream::_dst** [protected]**2.27 Bastei::Ipc_server Class Reference**

```
#include <ipc_generic.h>
```

Inheritance diagram for [Bastei::Ipc_server](#):

**Public Member Functions**

- [Ipc_server](#) ([Msgbuf_base](#) *snd_msg, [Msgbuf_base](#) *rcv_msg)
- [Ipc_server](#) & [operator](#)>> ([Ipc_server_reply_wait](#))
- void [ret](#) (int retval)
- template<typename T>
[Ipc_server](#) & [operator](#)<< (T value)

- `template<typename T>`
`Ipc_server & operator>> (T &value)`

Protected Member Functions

- `void _prepare_next_reply_wait ()`
- `void _reply_wait ()`

Protected Attributes

- `bool _reply_needed`

2.27.1 Constructor & Destructor Documentation

2.27.1.1 `Bastei::Ipc_server::Ipc_server (Msgbuf_base * snd_msg, Msgbuf_base * rcv_msg)`

Constructor

2.27.2 Member Function Documentation

2.27.2.1 `void Bastei::Ipc_server::_prepare_next_reply_wait ()` `[protected]`

2.27.2.2 `void Bastei::Ipc_server::_reply_wait ()` `[protected]`

Send result of previous RPC request and wait for new one

2.27.2.3 `Ipc_server& Bastei::Ipc_server::operator>> (Ipc_server_reply_wait)` `[inline]`

Reply current request and wait for a new one

2.27.2.4 `void Bastei::Ipc_server::ret (int retval)` `[inline]`

Set return value of server call

2.27.2.5 `template<typename T> Ipc_server& Bastei::Ipc_server::operator<< (T value)` `[inline]`

Insert value into send buffer

Reimplemented from [Bastei::Ipc_ostream](#).

2.27.2.6 `template<typename T> Ipc_server& Bastei::Ipc_server::operator>> (T & value)` `[inline]`

Read values from receive buffer

Reimplemented from [Bastei::Ipc_istream](#).

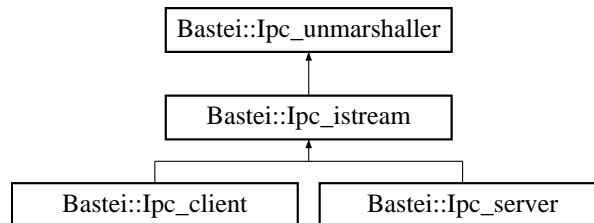
2.27.3 Member Data Documentation

2.27.3.1 `bool Bastei::Ipc_server::_reply_needed` [protected]

2.28 Bastei::Ipc_unmarshaller Class Reference

```
#include <ipc_generic.h>
```

Inheritance diagram for Bastei::Ipc_unmarshaller::



Public Member Functions

- [Ipc_unmarshaller](#) (char *rcvbuf, size_t rcvbuf_size)

Protected Member Functions

- `template<typename T>`
void [_read_from_buf](#) (T &value)
- void [_read_bytebuf_from_buf](#) (Buffer &b)
- long [_long_at_idx](#) (int idx)

Protected Attributes

- char * [_rcvbuf](#)
- size_t [_rcvbuf_size](#)
- unsigned [_read_offset](#)

2.28.1 Detailed Description

Unmarshal arguments from receive buffer

2.28.2 Constructor & Destructor Documentation

2.28.2.1 `Bastei::Ipc_unmarshaller::Ipc_unmarshaller` (char * *rcvbuf*, size_t *rcvbuf_size*) [inline]

2.28.3 Member Function Documentation

2.28.3.1 `template<typename T> void Bastei::lpc_unmarshaller::_read_from_buf (T & value)` [inline, protected]

Read value of type T from buffer

2.28.3.2 `void Bastei::lpc_unmarshaller::_read_bytebuf_from_buf (Buffer & b)` [inline, protected]

Read [Buffer](#) from receive buffer

2.28.3.3 `long Bastei::lpc_unmarshaller::_long_at_idx (int idx)` [inline, protected]

Read long value at specified byte index of receive buffer

2.28.4 Member Data Documentation

2.28.4.1 `char* Bastei::Ipc_unmarshaller::_rcvbuf` [protected]

2.28.4.2 `size_t Bastei::Ipc_unmarshaller::_rcvbuf_size` [protected]

2.28.4.3 `unsigned Bastei::Ipc_unmarshaller::_read_offset` [protected]

2.29 Bastei::Irq_session Class Reference

```
#include <irq_session.h>
```

Public Member Functions

- virtual `~Irq_session ()`
- virtual void `wait_for_irq ()=0`

Protected Types

- enum `Opcode { WAIT_FOR_IRQ }`

2.29.1 Member Enumeration Documentation

2.29.1.1 `enum Bastei::Irq_session::Opcode` [protected]

Enumerator:

`WAIT_FOR_IRQ`

2.29.2 Constructor & Destructor Documentation

2.29.2.1 virtual Bastei::Irq_session::~~Irq_session () [inline, virtual]

2.29.3 Member Function Documentation

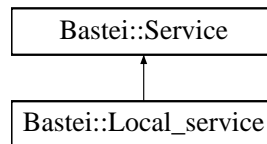
2.29.3.1 virtual void Bastei::Irq_session::wait_for_irq () [pure virtual]

Wait for interrupt occurrence

2.30 Bastei::Local_service Class Reference

```
#include <service.h>
```

Inheritance diagram for Bastei::Local_service::



Public Member Functions

- [Local_service](#) (const char *name, [Root](#) *root)
- [Root](#) * [root](#) ()

2.30.1 Detailed Description

Representation of a service that is implemented locally

2.30.2 Constructor & Destructor Documentation

2.30.2.1 Bastei::Local_service::Local_service (const char * *name*, [Root](#) * *root*) [inline]

2.30.3 Member Function Documentation

2.30.3.1 [Root](#)* Bastei::Local_service::root () [inline, virtual]

Implements [Bastei::Service](#).

2.31 Bastei::Lock Class Reference

```
#include <lock.h>
```

Public Types

- typedef `Lock_guard< Lock >` `Guard`

Public Member Functions

- `Lock` (State initial=UNLOCKED)
- void `lock` ()

2.31.1 Member Typedef Documentation

2.31.1.1 typedef `Lock_guard<Lock>` `Bastei::Lock::Guard`

`Lock` guard

2.31.2 Constructor & Destructor Documentation

2.31.2.1 `Bastei::Lock::Lock` (State *initial* = UNLOCKED) [`inline`, `explicit`]

Constructor

2.31.3 Member Function Documentation

2.31.3.1 void `Bastei::Lock::lock` () [`inline`]

2.32 `Bastei::Lock_guard< LT >` Class Template Reference

```
#include <lock_guard.h>
```

Public Member Functions

- `Lock_guard` (LT &lock)
- `~Lock_guard` ()

2.32.1 Detailed Description

```
template<typename LT> class Bastei::Lock_guard< LT >
```

`Lock` guard template

Parameters:

LT `Lock` type

2.32.2 Constructor & Destructor Documentation

2.32.2.1 template<typename LT> `Bastei::Lock_guard< LT >::Lock_guard` (LT & *lock*) [`inline`, `explicit`]

2.32.2.2 `template<typename LT> Bastei::Lock_guard< LT >::~~Lock_guard ()` [inline]

2.33 Bastei::Log_session Class Reference

```
#include <log_session.h>
```

Public Member Functions

- virtual `~Log_session ()`
- virtual `size_t write (const char *string)=0`

Protected Types

- enum `Opcode { WRITE }`

2.33.1 Member Enumeration Documentation

2.33.1.1 `enum Bastei::Log_session::Opcode` [protected]

Enumerator:

WRITE

2.33.2 Constructor & Destructor Documentation

2.33.2.1 `virtual Bastei::Log_session::~~Log_session ()` [inline, virtual]

2.33.3 Member Function Documentation

2.33.3.1 `virtual size_t Bastei::Log_session::write (const char * string)` [pure virtual]

Output null-terminated string

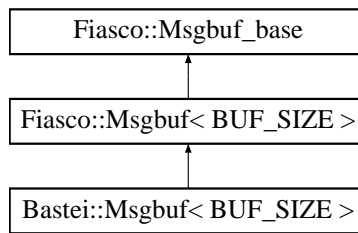
Returns:

Number of written characters

2.34 Fiasco::Msgbuf< BUF_SIZE > Class Template Reference

```
#include <ipc_msgbuf.h>
```

Inheritance diagram for `Fiasco::Msgbuf< BUF_SIZE >::`



Public Member Functions

- [Msgbuf \(\)](#)

Public Attributes

- char [buf](#) [BUF_SIZE]

2.34.1 Detailed Description

`template<unsigned BUF_SIZE> class Fiasco::Msgbuf< BUF_SIZE >`

Instance of IPC message buffer with specified buffer size

2.34.2 Constructor & Destructor Documentation

2.34.2.1 `template<unsigned BUF_SIZE> Fiasco::Msgbuf< BUF_SIZE >::Msgbuf ()`
 [inline]

2.34.3 Member Data Documentation

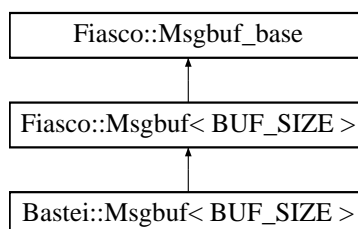
2.34.3.1 `template<unsigned BUF_SIZE> char Fiasco::Msgbuf< BUF_SIZE >::buf[BUF_SIZE]`

Reimplemented from [Fiasco::Msgbuf_base](#).

2.35 *Bastei::Msgbuf< BUF_SIZE > Class Template Reference*

```
#include <ipc_msgbuf.h>
```

Inheritance diagram for `Bastei::Msgbuf< BUF_SIZE >::`

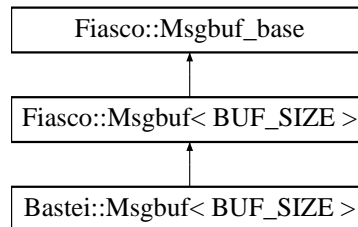


```
template<int BUF_SIZE> class Bastei::Msgbuf< BUF_SIZE >
```

2.36 Fiasco::Msgbuf_base Class Reference

```
#include <ipc_msgbuf.h>
```

Inheritance diagram for Fiasco::Msgbuf_base::



Public Member Functions

- Bastei::size_t [size \(\)](#)
- void * [addr \(\)](#)

Public Attributes

- l4_fpage_t [rcv_fpage](#)
- l4_msgdope_t [size_dope](#)
- l4_msgdope_t [send_dope](#)
- char [buf \[\]](#)

Protected Attributes

- Bastei::size_t [_size](#)
- char [_msg_start \[\]](#)

2.36.1 Detailed Description

IPC message buffer layout

2.36.2 Member Function Documentation

2.36.2.1 Bastei::size_t Fiasco::Msgbuf_base::size () [inline]

Return size of message buffer

2.36.2.2 void* Fiasco::Msgbuf_base::addr () [inline]

Return address of message buffer

2.36.3 Member Data Documentation

2.36.3.1 **Bastei::size_t** *Fiasco::Msgbuf_base::_size* [protected]

2.36.3.2 **char** *Fiasco::Msgbuf_base::_msg_start*[] [protected]

2.36.3.3 **l4_fpage_t** *Fiasco::Msgbuf_base::rcv_fpage*

2.36.3.4 **l4_msgdope_t** *Fiasco::Msgbuf_base::size_dope*

2.36.3.5 **l4_msgdope_t** *Fiasco::Msgbuf_base::send_dope*

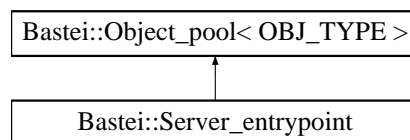
2.36.3.6 **char** *Fiasco::Msgbuf_base::buf*[]

Reimplemented in [Fiasco::Msgbuf](#)< BUF_SIZE >.

2.37 *Bastei::Object_pool*< OBJ_TYPE > Class Template Reference

```
#include <object_pool.h>
```

Inheritance diagram for *Bastei::Object_pool*< OBJ_TYPE >::



Public Member Functions

- void [insert](#) (OBJ_TYPE *obj)
- void [remove](#) (OBJ_TYPE *obj)
- OBJ_TYPE * [obj_by_id](#) (long obj_id)
- OBJ_TYPE * [obj_by_cap](#) ([Capability](#) cap)
- OBJ_TYPE * [first](#) ()

Classes

- class [Entry](#)

2.37.1 Detailed Description

template<typename OBJ_TYPE> **class** *Bastei::Object_pool*< OBJ_TYPE >

Map object ids to local objects

Parameters:

OBJ_TYPE Object type (must be inherited from Object_pool_entry)

The local names of a capabilities are used to differentiate multiple server objects managed by one and the same object pool.

2.37.2 Member Function Documentation

2.37.2.1 `template<typename OBJ_TYPE> void Bastei::Object_pool< OBJ_TYPE >::insert (OBJ_TYPE * obj)` [inline]

2.37.2.2 `template<typename OBJ_TYPE> void Bastei::Object_pool< OBJ_TYPE >::remove (OBJ_TYPE * obj)` [inline]

2.37.2.3 `template<typename OBJ_TYPE> OBJ_TYPE* Bastei::Object_pool< OBJ_TYPE >::obj_by_id (long obj_id)` [inline]

Lookup object

2.37.2.4 `template<typename OBJ_TYPE> OBJ_TYPE* Bastei::Object_pool< OBJ_TYPE >::obj_by_cap (Capability cap)` [inline]

2.37.2.5 `template<typename OBJ_TYPE> OBJ_TYPE* Bastei::Object_pool< OBJ_TYPE >::first ()` [inline]

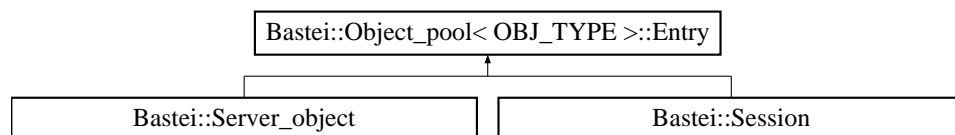
Return first element of tree

This function is used for removing tree elements step by step.

2.38 Bastei::Object_pool< OBJ_TYPE >::Entry Class Reference

```
#include <object_pool.h>
```

Inheritance diagram for Bastei::Object_pool< OBJ_TYPE >::Entry::

**Public Types**

- enum { `OBJ_ID_INVALID = 0` }

Public Member Functions

- [Entry](#) ()
- [Entry](#) ([Capability](#) cap)
- bool [higher](#) ([Entry](#) *s)
- [Entry](#) * [find_by_obj_id](#) (long obj_id)
- void [cap](#) ([Capability](#) c)
- [Capability](#) [cap](#) ()

Friends

- class [Object_pool](#)
- class [Avl_node](#)< [Entry](#) >

template<typename OBJ_TYPE> class **Bastei::Object_pool**< OBJ_TYPE >::Entry

2.38.1 Member Enumeration Documentation

2.38.1.1 template<typename OBJ_TYPE> anonymous enum

Enumerator:

OBJ_ID_INVALID

2.38.2 Constructor & Destructor Documentation

2.38.2.1 template<typename OBJ_TYPE> **Bastei::Object_pool**< OBJ_TYPE >::Entry::Entry () [inline]

Constructors

2.38.2.2 template<typename OBJ_TYPE> **Bastei::Object_pool**< OBJ_TYPE >::Entry::Entry ([Capability](#) cap) [inline]

2.38.3 Member Function Documentation

2.38.3.1 template<typename OBJ_TYPE> bool **Bastei::Object_pool**< OBJ_TYPE >::Entry::higher ([Entry](#) * s) [inline]

[Avl_node](#) interface

2.38.3.2 template<typename OBJ_TYPE> [Entry](#)* **Bastei::Object_pool**< OBJ_TYPE >::Entry::find_by_obj_id (long obj_id) [inline]

Support for object pool

2.38.3.3 `template<typename OBJ_TYPE> void Bastei::Object_pool< OBJ_TYPE >::Entry::cap (Capability c) [inline]`

Assign capability to object pool entry

2.38.3.4 `template<typename OBJ_TYPE> Capability Bastei::Object_pool< OBJ_TYPE >::Entry::cap () [inline]`

2.38.4 Friends And Related Function Documentation

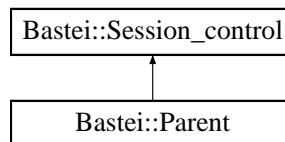
2.38.4.1 `template<typename OBJ_TYPE> friend class Object_pool [friend]`

2.38.4.2 `template<typename OBJ_TYPE> friend class Avl_node< Entry > [friend]`

2.39 Bastei::Parent Class Reference

```
#include <parent.h>
```

Inheritance diagram for Bastei::Parent::



Public Types

- enum { `MSGBUF_SIZE` = 256 }

Public Member Functions

- virtual `~Parent ()`
- virtual void `exit (int exit_value)=0`
- virtual int `announce (const char *service_name, Capability service_root)=0`
- virtual `Capability session (const char *service_name, const char *args)=0`
- virtual int `transfer_quota (Capability to_session_cap, size_t amount)=0`

Protected Types

- enum `Opcode` {
`EXIT, ANNOUNCE, SESSION, TRANSFER_QUOTA,`
`CLOSE }`

2.39.1 Member Enumeration Documentation

2.39.1.1 anonymous enum

Enumerator:

MSGBUF_SIZE

2.39.1.2 enum `Bastei::Parent::Opcode` [protected]

Enumerator:

EXIT

ANNOUNCE

SESSION

TRANSFER_QUOTA

CLOSE

2.39.2 Constructor & Destructor Documentation

2.39.2.1 virtual `Bastei::Parent::~~Parent ()` [inline, virtual]

2.39.3 Member Function Documentation

2.39.3.1 virtual void `Bastei::Parent::exit (int exit_value)` [pure virtual]

Tell parent to exit the program

2.39.3.2 virtual int `Bastei::Parent::announce (const char * service_name, Capability service_root)` [pure virtual]

Announce service to the parent

2.39.3.3 virtual Capability `Bastei::Parent::session (const char * service_name, const char * args)` [pure virtual]

Create session to a service

Parameters:

service_name name of the requested interface

args session constructor arguments

Returns:

capability to session (may be invalid if an error occurred)

The session id is used for referring the session at the parent. Direct communication with the session is performed via `out_session_cap`.

By convention, an argument value for 'ram_quota' must be specified in 'args'. The specified amount of RAM will be donated to the session-providing service.

2.39.3.4 virtual int Bastei::Parent::transfer_quota (Capability *to_session_cap*, *size_t amount*) [pure virtual]

Transfer our quota to the server that provides the specified session

Parameters:

- to_session_cap* Recipient session.
- amount* Quota (in bytes) to transfer.

Returns:

0 if quota could be transferred or -1 on error

The error case means that there is not enough unused quota on the source side.

2.40 Bastei::Ram_dataspac Class Reference

```
#include <ram_dataspac.h>
```

Public Member Functions

- [Ram_dataspac](#) ([Ram_session](#) *ram_session, [size_t](#) size)
- [~Ram_dataspac](#) ()
- [Capability cap](#) ()
- [template<typename T>](#)
T * [local_addr](#) ()

2.40.1 Constructor & Destructor Documentation

2.40.1.1 Bastei::Ram_dataspac::Ram_dataspac ([Ram_session](#) * *ram_session*, [size_t size](#)) [inline]

Constructor

Exceptions:

- [Ram_session::Alloc_failed](#)
- [Rm_session::Attach_failed](#)

2.40.1.2 Bastei::Ram_dataspac::~~Ram_dataspac () [inline]

Destructor

2.40.2 Member Function Documentation

2.40.2.1 Capability Bastei::Ram_dataspac::cap () [inline]

Return capability of the used RAM dataspac

2.40.2.2 `template<typename T> T* Bastei::Ram_dataspace::local_addr () [inline]`

Request local address

This is a template to avoid inconvenient casts at the caller. A newly allocated RAM dataspace is untyped memory anyway.

2.41 *Bastei::Ram_session Class Reference*

```
#include <ram_session.h>
```

Public Member Functions

- virtual `~Ram_session ()`
- virtual `Capability alloc (size_t size)=0`
- virtual void `free (Capability ds_cap)=0`
- virtual int `ref_account (Capability ram_session)=0`
- virtual int `transfer_quota (Capability ram_session, size_t amount)=0`
- virtual `size_t quota ()=0`
- virtual `size_t used ()=0`
- `size_t avail ()`

Protected Types

- enum `Opcode {
 ALLOC, FREE, REF_ACCOUNT, TRANSFER_QUOTA,
 QUOTA, USED }`

Classes

- class `Alloc_failed`
- class `Quota_exceeded`

2.41.1 Member Enumeration Documentation

2.41.1.1 `enum Bastei::Ram_session::Opcode [protected]`

Enumerator:

ALLOC

FREE

REF_ACCOUNT

TRANSFER_QUOTA

QUOTA

USED

2.41.2 Constructor & Destructor Documentation

2.41.2.1 virtual Bastei::Ram_session::~~Ram_session () [inline, virtual]

Destructor

2.41.3 Member Function Documentation

2.41.3.1 virtual Capability Bastei::Ram_session::alloc (size_t size) [pure virtual]

Allocate RAM dataspace

Parameters:

size Size of RAM dataspace

Returns:

Capability to new RAM dataspace

Exceptions:

[*Quota_exceeded*](#)

2.41.3.2 virtual void Bastei::Ram_session::free (Capability ds_cap) [pure virtual]

Free RAM dataspace

Parameters:

ds_cap [Dataspac](#)e capability as returned by alloc

2.41.3.3 virtual int Bastei::Ram_session::ref_account (Capability ram_session) [pure virtual]

Define reference account for the RAM session

Parameters:

ram_session Reference account

Returns:

0 on success

Each RAM session requires another RAM session as reference account to transfer quota to and from. The reference account can be defined only once.

2.41.3.4 virtual int Bastei::Ram_session::transfer_quota (Capability ram_session, size_t amount) [pure virtual]

Transfer quota the another ram session

Parameters:

ram_session Receiver of quota donation

amount Amount of quota to donate

Returns:

0 on success

Quota can only be transferred if the specified RAM session is either the reference account for this session or vice versa.

2.41.3.5 virtual size_t Bastei::Ram_session::quota () [pure virtual]

Return current quota limit

2.41.3.6 virtual size_t Bastei::Ram_session::used () [pure virtual]

Return used quota

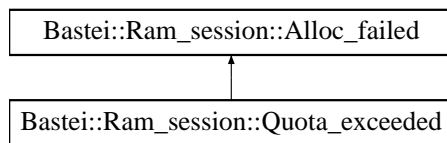
2.41.3.7 size_t Bastei::Ram_session::avail () [inline]

Return amount of available quota

2.42 Bastei::Ram_session::Alloc_failed Class Reference

```
#include <ram_session.h>
```

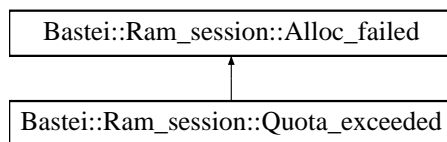
Inheritance diagram for Bastei::Ram_session::Alloc_failed::



2.43 Bastei::Ram_session::Quota_exceeded Class Reference

```
#include <ram_session.h>
```

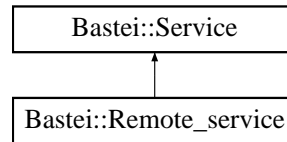
Inheritance diagram for Bastei::Ram_session::Quota_exceeded::



2.44 Bastei::Remote_service Class Reference

```
#include <service.h>
```

Inheritance diagram for Bastei::Remote_service::



Public Member Functions

- [Remote_service](#) (const char *name, [Capability](#) root_cap, [Child](#) *server)
- [Root](#) * root ()
- [Child](#) * server ()

2.44.1 Detailed Description

Representation of a service that is implemented in a child

2.44.2 Constructor & Destructor Documentation

2.44.2.1 Bastei::Remote_service::Remote_service (const char * *name*, [Capability](#) *root_cap*, [Child](#) * *server*) [inline]

Constructor

Parameters:

- name*** Name of service
- root_cap*** [Capability](#) to root interface
- server*** [Child](#) that provides the service

2.44.3 Member Function Documentation

2.44.3.1 [Root](#)* Bastei::Remote_service::root () [inline, virtual]

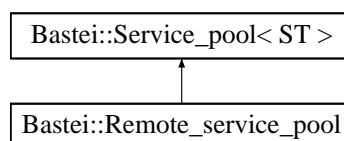
Implements [Bastei::Service](#).

2.44.3.2 [Child](#)* Bastei::Remote_service::server () [inline]

2.45 Bastei::Remote_service_pool Class Reference

```
#include <service.h>
```

Inheritance diagram for Bastei::Remote_service_pool::



Public Member Functions

- [Remote_service](#) * [find_by_server](#) ([Child](#) *server)

2.45.1 Member Function Documentation

2.45.1.1 Remote_service* [Bastei::Remote_service_pool::find_by_server](#) ([Child](#) * *server*) [inline]

Return first service provided by specified server

2.46 [Bastei::Rm_session Class Reference](#)

```
#include <rm_session.h>
```

Public Member Functions

- virtual [~Rm_session](#) ()
- virtual void * [attach](#) ([Capability](#) ds_cap, size_t size=0, off_t offset=0, addr_t local_addr=0)=0
- virtual void [detach](#) (void *local_addr)=0
- virtual int [add_client](#) ([Capability](#) thread_cap, [Capability](#) *pager_cap)=0

Protected Types

- enum [Opcode](#) { [ATTACH](#), [DETACH](#), [ADD_CLIENT](#) }

Classes

- class [Attach_failed](#)
- class [Invalid_dataspace](#)
- class [Region_conflict](#)

2.46.1 Member Enumeration Documentation

2.46.1.1 enum [Bastei::Rm_session::Opcode](#) [protected]

Enumerator:

ATTACH

DETACH

ADD_CLIENT

2.46.2 Constructor & Destructor Documentation

2.46.2.1 virtual [Bastei::Rm_session::~Rm_session](#) () [inline, virtual]

Destructor

2.46.3 Member Function Documentation

2.46.3.1 virtual void* Bastei::Rm_session::attach (Capability *ds_cap*, size_t *size* = 0, off_t *offset* = 0, addr_t *local_addr* = 0) [pure virtual]

Map dataspace into local address space

Parameters:

ds_cap Capability of dataspace to map

size Size of the locally mapped region default (0) is the whole dataspace

offset Start at offset in dataspace

local_addr Local destination address

Returns:

Local address of mapped dataspace

Exceptions:

Attach_failed if dataspace is invalid or on region conflict

2.46.3.2 virtual void Bastei::Rm_session::detach (void * *local_addr*) [pure virtual]

Remove region from local address space

2.46.3.3 virtual int Bastei::Rm_session::add_client (Capability *thread_cap*, Capability * *pager_cap*) [pure virtual]

Add client to pager

Parameters:

thread_cap **Thread** that will be paged

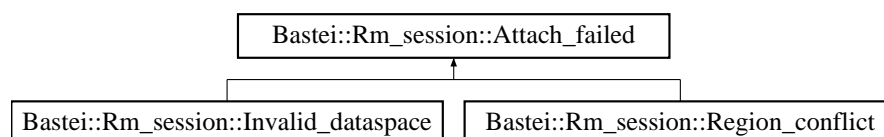
pager_cap Capability to be used for page faults

This method must be called at least once to establish a valid communication channel between the pager part of the region manager and the client thread.

2.47 Bastei::Rm_session::Attach_failed Class Reference

```
#include <rm_session.h>
```

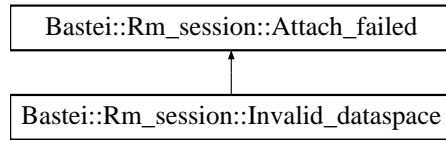
Inheritance diagram for Bastei::Rm_session::Attach_failed::



2.48 Bastei::Rm_session::Invalid_datspace Class Reference

```
#include <rm_session.h>
```

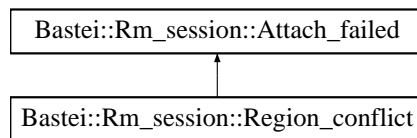
Inheritance diagram for Bastei::Rm_session::Invalid_datspace::



2.49 Bastei::Rm_session::Region_conflict Class Reference

```
#include <rm_session.h>
```

Inheritance diagram for Bastei::Rm_session::Region_conflict::



2.50 Bastei::Rom_session Class Reference

```
#include <rom_session.h>
```

Public Member Functions

- virtual [~Rom_session](#) ()
- virtual [Capability datspace](#) ()=0

Protected Types

- enum [Opcode](#) { [DATASPACE](#) }

2.50.1 Member Enumeration Documentation

2.50.1.1 enum Bastei::Rom_session::Opcode [protected]

Enumerator:

DATASPACE

2.50.2 Constructor & Destructor Documentation

2.50.2.1 virtual Bastei::Rom_session::~~Rom_session () [inline, virtual]

2.50.3 Member Function Documentation

2.50.3.1 virtual Capability Bastei::Rom_session::dataspace () [pure virtual]

Request dataspace containing the ROM session data

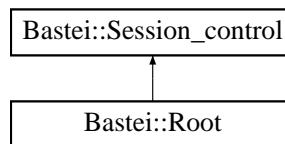
Returns:

Capability to ROM dataspace The capability may be invalid.

2.51 Bastei::Root Class Reference

```
#include <root.h>
```

Inheritance diagram for Bastei::Root::



Public Member Functions

- virtual [~Root](#) ()
- virtual [Capability session](#) (const char *args)=0

Protected Types

- enum [Opcode](#) { [SESSION](#), [CLOSE](#) }

2.51.1 Member Enumeration Documentation

2.51.1.1 enum Bastei::Root::Opcode [protected]

Enumerator:

SESSION

CLOSE

2.51.2 Constructor & Destructor Documentation

2.51.2.1 virtual Bastei::Root::~~Root () [inline, virtual]

2.51.3 Member Function Documentation

2.51.3.1 virtual Capability Bastei::Root::session (const char * args) [pure virtual]

Create session

Returns:

Capability to new session (may be invalid)

2.52 Bastei::Semaphore Class Reference

```
#include <semaphore.h>
```

Public Member Functions

- Semaphore ()
- void up ()
- void down ()

2.52.1 Constructor & Destructor Documentation

2.52.1.1 Bastei::Semaphore::Semaphore () [inline]

2.52.2 Member Function Documentation

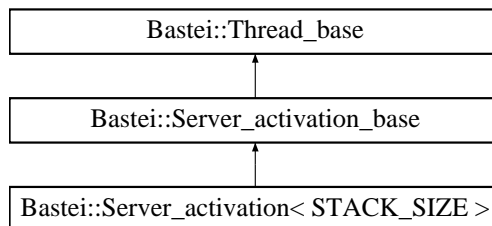
2.52.2.1 void Bastei::Semaphore::up () [inline]

2.52.2.2 void Bastei::Semaphore::down () [inline]

2.53 Bastei::Server_activation< STACK_SIZE > Class Template Reference

```
#include <server.h>
```

Inheritance diagram for Bastei::Server_activation< STACK_SIZE >::



Public Member Functions

- void start ()
- Server_activation (Msgbuf_base *snd_msg, Msgbuf_base *rcv_msg, bool start_on_construction=true, const char *name="activation")

```
template<int STACK_SIZE> class Bastei::Server_activation< STACK_SIZE >
```

2.53.1 Constructor & Destructor Documentation

2.53.1.1 `template<int STACK_SIZE> Bastei::Server_activation< STACK_SIZE >::Server_activation (Msgbuf_base * snd_msg, Msgbuf_base * rcv_msg, bool start_on_construction = true, const char * name = "activation") [inline]`

Constructor

Parameters:

snd_msg Message buffer for send part of IPC

rcv_msg Message buffer for receive part of IPC

start_on_construction Specifies whether the activation thread should be started immediately (default) or explicitly via the [start\(\)](#) function.

name Name of server activation thread (used for debugging)

The server activation thread will initialize its capability and but will not immediately enable the processing of requests. This way, the activation- using server can ensure that it gets initialized completely before the first capability invokations come in. Once the server is ready, it must enable the server activation explicitly by calling the [start\(\)](#) function. The 'start_on_construction' argument is a shortcut for the common case where the server's capability is handed over to other parties *_after_* the server is completely initialized.

2.53.2 Member Function Documentation

2.53.2.1 `template<int STACK_SIZE> void Bastei::Server_activation< STACK_SIZE >::start () [inline]`

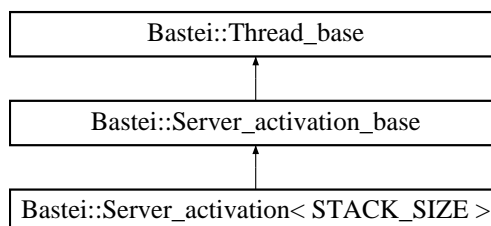
Unlock server activation if it is ready to serve requests

This function must be called before using the activation, for example, by supplying the activation to an endpoint.

2.54 Bastei::Server_activation_base Class Reference

```
#include <server.h>
```

Inheritance diagram for Bastei::Server_activation_base:



Public Member Functions

- void [ep](#) ([Server_entrypoint](#) *ep)
- void [entry](#) ()
- [Capability](#) [cap](#) ()
- void [leave_server_object](#) ([Server_object](#) *obj)

Protected Member Functions

- [Server_activation_base](#) ([Msgbuf_base](#) *snd_msg, [Msgbuf_base](#) *rcv_msg)

Protected Attributes

- [Server_object](#) * [_curr_obj](#)
- [Lock_curr_obj_lock](#)
- [Lock_cap_valid](#)
- [Lock_delay_start](#)

2.54.1 Constructor & Destructor Documentation

2.54.1.1 [Bastei::Server_activation_base::Server_activation_base](#) ([Msgbuf_base](#) * *snd_msg*, [Msgbuf_base](#) * *rcv_msg*) [`inline`, `protected`]

2.54.2 Member Function Documentation

2.54.2.1 void [Bastei::Server_activation_base::ep](#) ([Server_entrypoint](#) * *ep*) [`inline`]

Set entrypoint which the activation serves

This function must be called by an Entrypoint on `add_activation()` only.

2.54.2.2 void [Bastei::Server_activation_base::entry](#) () [`virtual`]

[Thread](#) interface

Implements [Bastei::Thread_base](#).

2.54.2.3 Capability [Bastei::Server_activation_base::cap](#) () [`inline`]

Return capability to this activation

This function should only be called from [Server_entrypoint](#)

2.54.2.4 void [Bastei::Server_activation_base::leave_server_object](#) ([Server_object](#) * *obj*)

Force activation to cancel dispatching the specified server object

2.54.3 Member Data Documentation

2.54.3.1 `Server_object*` `Bastei::Server_activation_base::_curr_obj` [protected]

2.54.3.2 Lock `Bastei::Server_activation_base::_curr_obj_lock` [protected]

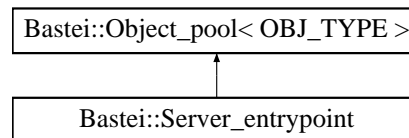
2.54.3.3 Lock `Bastei::Server_activation_base::_cap_valid` [protected]

2.54.3.4 Lock `Bastei::Server_activation_base::_delay_start` [protected]

2.55 Bastei::Server_entrypoint Class Reference

```
#include <server.h>
```

Inheritance diagram for `Bastei::Server_entrypoint`:



Public Member Functions

- `Server_entrypoint` (`Cap_session *cap_session`, `Server_activation_base *a=0`)
- `int add_activation` (`Server_activation_base *t`)
- `Capability manage` (`Server_object *obj`)
- `void dissolve` (`Server_object *obj`)

2.55.1 Detailed Description

Entrypoint into a server

For L4v2 and Linux, a `Server_entrypoint` can hold only one activation and thus, allow only serial access to the `Server_objects` managed by this endpoint. For a multithreaded server, multiple entrypoints (one per thread) have to be created.

2.55.2 Constructor & Destructor Documentation

2.55.2.1 `Bastei::Server_entrypoint::Server_entrypoint` (`Cap_session * cap_session`, `Server_activation_base * a = 0`)

Constructor

Parameters:

`cap_session` `Cap_session` for creating capabilities for the server objects managed by this entrypoint.

a Initial activation

2.55.3 Member Function Documentation

2.55.3.1 `int Bastei::Server_entrypoint::add_activation (Server_activation_base * f)`

Add activation to activation pool of this endpoint

2.55.3.2 Capability `Bastei::Server_entrypoint::manage (Server_object * obj)`

Associate [Server_object](#) with the endpoint

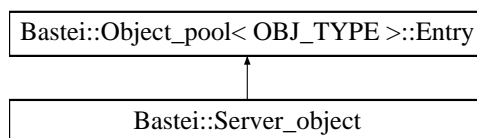
2.55.3.3 `void Bastei::Server_entrypoint::dissolve (Server_object * obj)`

Dissolve [Server_object](#) from endpoint

2.56 *Bastei::Server_object Class Reference*

```
#include <server.h>
```

Inheritance diagram for `Bastei::Server_object`:



Public Member Functions

- virtual `~Server_object ()`
- void `lock ()`
- void `unlock ()`
- virtual int `dispatch (int op, Ipc_istream &is, Ipc_ostream &os)=0`

2.56.1 Detailed Description

A [Server_object](#) is a locally implemented object that can be referenced from the outer world using a capability. The capability gets created when attaching a [Server_object](#) to an [Server_entrypoint](#).

2.56.2 Constructor & Destructor Documentation

2.56.2.1 `virtual Bastei::Server_object::~~Server_object ()` [`inline`, `virtual`]

2.56.3 Member Function Documentation

2.56.3.1 `void Bastei::Server_object::lock ()` [`inline`]

2.56.3.2 void Bastei::Server_object::unlock () [inline]

2.56.3.3 virtual int Bastei::Server_object::dispatch (int *op*, Ipc_istream & *is*, Ipc_ostream & *os*) [pure virtual]

Interface to be implemented by a derived class

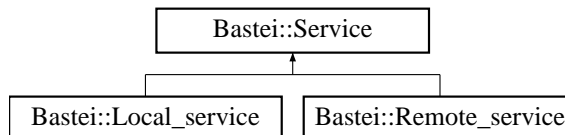
Parameters:

- op*** opcode of invoked method
- is*** Ipc_input stream with method arguments
- os*** Ipc_output stream for storing method results

2.57 Bastei::Service Class Reference

```
#include <service.h>
```

Inheritance diagram for Bastei::Service::



Public Member Functions

- virtual [~Service](#) ()
- [Service](#) (const char *name)
- virtual [Root](#) * [root](#) ()=0
- const char * [name](#) ()

2.57.1 Constructor & Destructor Documentation

2.57.1.1 virtual Bastei::Service::~~Service () [inline, virtual]

2.57.1.2 Bastei::Service::Service (const char * *name*) [inline]

2.57.2 Member Function Documentation

2.57.2.1 virtual Root* Bastei::Service::root () [pure virtual]

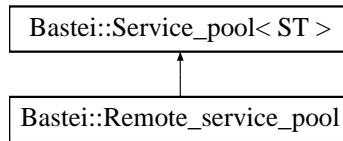
Implemented in [Bastei::Local_service](#), and [Bastei::Remote_service](#).

2.57.2.2 const char* Bastei::Service::name () [inline]

2.58 **Bastei::Service_pool< ST > Class Template Reference**

```
#include <service.h>
```

Inheritance diagram for `Bastei::Service_pool< ST >::`



Public Member Functions

- `ST * find` (const char *name)
- void `dump_service_wait_queue` ()
- `ST * wait_for_service` (const char *name, `Client *client`)
- void `insert` (ST *service)
- void `remove` (ST *service)

Protected Attributes

- `Lock_service_wait_queue_lock`
- `List< Client > _service_wait_queue`
- `List< ST > _services`

2.58.1 Detailed Description

```
template<typename ST> class Bastei::Service_pool< ST >
```

Container for holding service representations

2.58.2 Member Function Documentation

2.58.2.1 `template<typename ST> ST* Bastei::Service_pool< ST >::find (const char * name)` [inline]

Probe service with specified name

2.58.2.2 `template<typename ST> void Bastei::Service_pool< ST >::dump_service_wait_queue ()` [inline]

2.58.2.3 `template<typename ST> ST* Bastei::Service_pool< ST >::wait_for_service (const char * name, Client * client)` [inline]

Wait for service

This function is called by the clients's thread when requesting a session creation. It blocks if the requested service is not available.

Returns:

[Service](#) structure that matches the request or 0 if the waiting was canceled.

2.58.2.4 `template<typename ST> void Bastei::Service_pool< ST >::insert (ST * service)`
[inline]

Register service

This function is called by the server's thread.

2.58.2.5 `template<typename ST> void Bastei::Service_pool< ST >::remove (ST * service)`
[inline]

Unregister service

2.58.3 Member Data Documentation

2.58.3.1 `template<typename ST> Lock Bastei::Service_pool< ST >::_service_wait_queue_`
`lock` [protected]

2.58.3.2 `template<typename ST> List<Client> Bastei::Service_pool< ST >::_service_`
`wait_queue` [protected]

2.58.3.3 `template<typename ST> List<ST> Bastei::Service_pool< ST >::_services`
[protected]

2.59 Nitpicker::Session Class Reference

```
#include <nitpicker_session.h>
```

Public Member Functions

- virtual `~Session ()`
- virtual `Bastei::Capability framebuffer_session ()=0`
- virtual `Bastei::Capability input_session ()=0`
- virtual `Bastei::Capability create_view ()=0`
- virtual void `destroy_view (Bastei::Capability view_cap)=0`
- virtual int `background (Bastei::Capability view_cap)=0`

Protected Types

- enum `Opcode` {
 `FRAMEBUFFER_SESSION`, `INPUT_SESSION`, `CREATE_VIEW`, `DESTROY_VIEW`,
 `BACKGROUND` }

2.59.1 Member Enumeration Documentation

2.59.1.1 enum `Nitpicker::Session::Opcode` [protected]

Enumerator:

FRAMEBUFFER_SESSION

INPUT_SESSION

CREATE_VIEW

DESTROY_VIEW

BACKGROUND

2.59.2 Constructor & Destructor Documentation

2.59.2.1 virtual `Nitpicker::Session::~~Session ()` [inline, virtual]

2.59.3 Member Function Documentation

2.59.3.1 virtual `Bastei::Capability Nitpicker::Session::framebuffer_session ()` [pure virtual]

Request framebuffer sub-session

2.59.3.2 virtual `Bastei::Capability Nitpicker::Session::input_session ()` [pure virtual]

Request input sub-session

2.59.3.3 virtual `Bastei::Capability Nitpicker::Session::create_view ()` [pure virtual]

Create a new view at the buffer

Returns:

Capability to a new view.

2.59.3.4 virtual void `Nitpicker::Session::destroy_view (Bastei::Capability view_cap)` [pure virtual]

Destroy view

2.59.3.5 virtual int Nitpicker::Session::background (Bastei::Capability *view_cap*) [pure virtual]

Define view that is used as desktop background

2.60 Pci::Session Class Reference

```
#include <pci_session.h>
```

Public Member Functions

- virtual [~Session](#) ()
- virtual [Bastei::Capability find_device_by_vendor](#) (unsigned short vendor_id, [Bastei::Capability prev_device](#))=0
- virtual void [release_device](#) ([Bastei::Capability device_cap](#))=0

Protected Types

- enum [Opcode](#) { [FIND_DEVICE_BY_VENDOR](#), [RELEASE_DEVICE](#) }

2.60.1 Member Enumeration Documentation

2.60.1.1 enum Pci::Session::Opcode [protected]

Enumerator:

FIND_DEVICE_BY_VENDOR

RELEASE_DEVICE

2.60.2 Constructor & Destructor Documentation

2.60.2.1 virtual Pci::Session::~~Session () [inline, virtual]

2.60.3 Member Function Documentation

2.60.3.1 virtual Bastei::Capability Pci::Session::find_device_by_vendor (unsigned short *vendor_id*, Bastei::Capability *prev_device*) [pure virtual]

Find device by the specified vendor ID

Parameters:

vendor_id Vendor ID matching the device

prev_device Previous device

The 'prev_device' argument can be used to iterate through the list of devices of the same vendor. For finding the first device, an invalid capability must be specified.

2.60.3.2 virtual void `Pci::Session::release_device (Bastei::Capability device_cap)` [pure virtual]

Free server-internal data structures representing the device

Use this function to relax the heap partition of your PCI session.

2.61 **Input::Session Class Reference**

```
#include <input_session.h>
```

Public Member Functions

- virtual `~Session ()`
- virtual `Bastei::Capability dataspace ()=0`
- virtual `bool is_pending ()=0`
- virtual `int flush ()=0`

Protected Types

- enum `Opcode { DATASPACE, IS_PENDING, FLUSH }`

2.61.1 Member Enumeration Documentation

2.61.1.1 enum `Input::Session::Opcode` [protected]

Enumerator:

DATASPACE
IS_PENDING
FLUSH

2.61.2 Constructor & Destructor Documentation

2.61.2.1 virtual `Input::Session::~~Session ()` [inline, virtual]

2.61.3 Member Function Documentation

2.61.3.1 virtual `Bastei::Capability Input::Session::dataspace ()` [pure virtual]

Return capability to event buffer dataspace

2.61.3.2 virtual `bool Input::Session::is_pending ()` [pure virtual]

Request input state

Returns:

True if new events are available

2.61.3.3 virtual int Input::Session::flush () [pure virtual]

Flush pending events to event buffer

Returns:

Number of flushed events

2.62 Framebuffer::Session Class Reference

```
#include <framebuffer_session.h>
```

Public Types

- enum [Mode](#) { [INVALID](#), [RGB565](#) }

Public Member Functions

- virtual [~Session](#) ()
- virtual [Bastei::Capability dataspace](#) ()=0
- virtual void [info](#) (int *out_w, int *out_h, [Mode](#) *out_mode)=0
- virtual void [refresh](#) (int x, int y, int w, int h)=0

Protected Types

- enum [Opcode](#) { [DATASPACE](#), [INFO](#), [REFRESH](#) }

2.62.1 Member Enumeration Documentation

2.62.1.1 enum Framebuffer::Session::Opcode [protected]

Enumerator:

DATASPACE

INFO

REFRESH

2.62.1.2 enum Framebuffer::Session::Mode

Pixel formats

Enumerator:

INVALID

RGB565

2.62.2 Constructor & Destructor Documentation

2.62.2.1 virtual Framebuffer::Session::~~Session () [inline, virtual]

2.62.3 Member Function Documentation

2.62.3.1 virtual Bastei::Capability Framebuffer::Session::dataspace () [pure virtual]

Request dataspace representing the logical frame buffer

2.62.3.2 virtual void Framebuffer::Session::info (int * *out_w*, int * *out_h*, Mode * *out_mode*) [pure virtual]

Request current screen mode properties

Parameters:

out_w Width of frame buffer

out_h Height of frame buffer

out_mode Pixel format

2.62.3.3 virtual void Framebuffer::Session::refresh (int *x*, int *y*, int *w*, int *h*) [pure virtual]

Flush specified pixel region

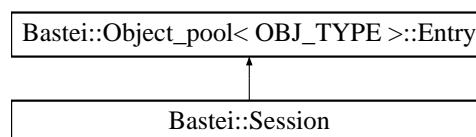
Parameters:

x,y,w,h Region to be updated on physical frame buffer

2.63 Bastei::Session Class Reference

```
#include <child.h>
```

Inheritance diagram for Bastei::Session::



Public Member Functions

- [Session](#) ([Capability](#) session_cap, [Session_control](#) *sc, size_t ram_quota, [Child](#) *server=0, const char *ident="<noname>")
- [Session](#) ()
- [Capability](#) session_cap ()
- [Session_control](#) * session_control ()
- size_t donated_ram_quota ()
- bool valid ()
- [Child](#) * server ()
- const char * ident ()

2.63.1 Constructor & Destructor Documentation

2.63.1.1 Bastei::Session::Session (Capability *session_cap*, Session_control * *sc*, size_t *ram_quota*, Child * *server* = 0, const char * *ident* = "<noname>") [inline]

Constructors

2.63.1.2 Bastei::Session::Session () [inline]

Default constructor creates invalid session

2.63.2 Member Function Documentation

2.63.2.1 Capability Bastei::Session::session_cap () [inline]

Accessors

2.63.2.2 Session_control* Bastei::Session::session_control () [inline]

2.63.2.3 size_t Bastei::Session::donated_ram_quota () [inline]

2.63.2.4 bool Bastei::Session::valid () [inline]

2.63.2.5 Child* Bastei::Session::server () [inline]

2.63.2.6 const char* Bastei::Session::ident () [inline]

2.64 Timer::Session Class Reference

```
#include <timer_session.h>
```

Public Member Functions

- virtual [~Session](#) ()
- virtual void [msleep](#) (unsigned ms)=0

Protected Types

- enum [Opcode](#) { [MSLEEP](#) }

2.64.1 Member Enumeration Documentation

2.64.1.1 enum `Timer::Session::Opcode` [protected]

Enumerator:

MSLEEP

2.64.2 Constructor & Destructor Documentation

2.64.2.1 virtual `Timer::Session::~~Session ()` [inline, virtual]

2.64.3 Member Function Documentation

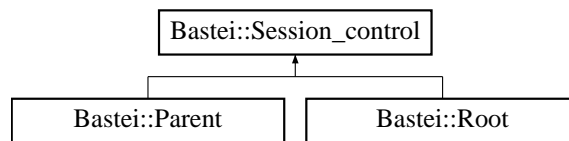
2.64.3.1 virtual void `Timer::Session::msleep (unsigned ms)` [pure virtual]

Sleep number of milliseconds

2.65 `Bastei::Session_control` Class Reference

```
#include <session_control.h>
```

Inheritance diagram for `Bastei::Session_control`:



Public Member Functions

- virtual `~Session_control ()`
- virtual void `close (Capability session_cap)=0`

2.65.1 Constructor & Destructor Documentation

2.65.1.1 virtual `Bastei::Session_control::~~Session_control ()` [inline, virtual]

2.65.2 Member Function Documentation

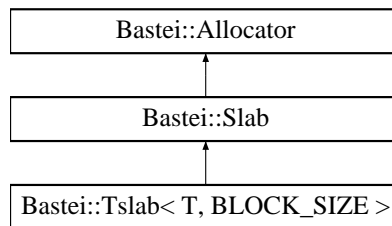
2.65.2.1 virtual void `Bastei::Session_control::close (Capability session_cap)` [pure virtual]

Close session

2.66 Bastei::Slab Class Reference

```
#include <slab.h>
```

Inheritance diagram for Bastei::Slab::



Public Member Functions

- size_t [slab_size](#) ()
- size_t [block_size](#) ()
- size_t [num_elem](#) ()
- size_t [entry_size](#) ()
- [Slab](#) (size_t slab_size, size_t block_size, [Slab_block](#) *initial_sb, [Allocator](#) *backingstore=0)
- [~Slab](#) ()
- void [dump_sb_list](#) ()
- void [remove_sb](#) ([Slab_block](#) *sb)
- void [insert_sb](#) ([Slab_block](#) *sb, [Slab_block](#) *at=0)
- void * [alloc](#) ()
- void * [first_used_elem](#) ()
- bool [num_free_entries_higher_than](#) (int n)
- void [backingstore](#) ([Allocator](#) *bs)
- [Allocator](#) * [backingstore](#) ()
- void * [alloc](#) (size_t)
- void [free](#) (void *addr, size_t)
- size_t [consumed](#) ()
- size_t [overhead](#) ()

Static Public Member Functions

- static void [free](#) (void *addr)

2.66.1 Detailed Description

[Slab](#) allocator

2.66.2 Constructor & Destructor Documentation

2.66.2.1 Bastei::Slab::Slab (size_t *slab_size*, size_t *block_size*, Slab_block * *initial_sb*, Allocator * *backingstore* = 0)

Constructor

At construction time, there exists one initial slab block that is used for the first couple of allocations, especially for the allocation of the second slab block.

2.66.2.2 Bastei::Slab::~~Slab ()

Destructor

2.66.3 Member Function Documentation

2.66.3.1 size_t Bastei::Slab::slab_size () [inline]

2.66.3.2 size_t Bastei::Slab::block_size () [inline]

2.66.3.3 size_t Bastei::Slab::num_elem () [inline]

2.66.3.4 size_t Bastei::Slab::entry_size () [inline]

2.66.3.5 void Bastei::Slab::dump_sb_list ()

Debug function for dumping the current slab block list

2.66.3.6 void Bastei::Slab::remove_sb (Slab_block * *sb*)

Remove block from slab block list

2.66.3.7 void Bastei::Slab::insert_sb (Slab_block * *sb*, Slab_block * *at* = 0)

Insert block into slab block list

2.66.3.8 void* Bastei::Slab::alloc ()

Allocate slab entry

2.66.3.9 static void Bastei::Slab::free (void * *addr*) [static]

Free slab entry

2.66.3.10 void* Bastei::Slab::first_used_elem ()

Return a used slab element

2.66.3.11 bool Bastei::Slab::num_free_entries_higher_than (int *n*)

Return true if number of free slab entries is higher than *n*

2.66.3.12 void Bastei::Slab::backingstore (Allocator * *bs*) [inline]

Define/request backingstore allocator

2.66.3.13 Allocator* Bastei::Slab::backingstore () [inline]**2.66.3.14 void* Bastei::Slab::alloc (size_t) [inline, virtual]**

[Allocator](#) interface

Implements [Bastei::Allocator](#).

2.66.3.15 void Bastei::Slab::free (void * *addr*, size_t *size*) [inline, virtual]

Free block a previously allocated block

Implements [Bastei::Allocator](#).

2.66.3.16 size_t Bastei::Slab::consumed () [virtual]

Return total amount of backingstore consumed by the allocator

Reimplemented from [Bastei::Allocator](#).

2.66.3.17 size_t Bastei::Slab::overhead () [inline, virtual]

Return metadata overhead per block

Implements [Bastei::Allocator](#).

2.67 Bastei::Slab_block Class Reference

```
#include <slab.h>
```

Public Member Functions

- [Slab_block](#) (Slab **s*=0)
- void [slab](#) (Slab **slab*)
- unsigned [avail](#) ()
- void * [alloc](#) ()
- [Slab_entry](#) * [first_used_entry](#) ()
- void [inc_avail](#) ([Slab_entry](#) **e*)
- void [dec_avail](#) ()
- void [dump](#) ()
- int [check_bounds](#) ()

Public Attributes

- [Slab_block * next](#)
- [Slab_block * prev](#)

2.67.1 Detailed Description

A slab block holds an array of slab entries.

2.67.2 Constructor & Destructor Documentation

2.67.2.1 **Bastei::Slab_block::Slab_block (Slab * *s* = 0)** [*inline*, *explicit*]

Constructor

Normally, Slab_blocks are constructed by a [Slab](#) allocator that specifies itself as constructor argument.

2.67.3 Member Function Documentation

2.67.3.1 **void Bastei::Slab_block::slab (Slab * *slab*)**

Configure block to be managed by the specified slab allocator

2.67.3.2 **unsigned Bastei::Slab_block::avail ()** [*inline*]

Request number of available entries in block

2.67.3.3 **void* Bastei::Slab_block::alloc ()**

Allocate slab entry from block

2.67.3.4 **Slab_entry* Bastei::Slab_block::first_used_entry ()**

Return a used slab block entry

2.67.3.5 **void Bastei::Slab_block::inc_avail (Slab_entry * *e*)**

These functions are called by [Slab_entry](#).

2.67.3.6 **void Bastei::Slab_block::dec_avail ()**

2.67.3.7 **void Bastei::Slab_block::dump ()**

Debug and test hooks

2.67.3.8 **int Bastei::Slab_block::check_bounds ()**

2.67.4 Member Data Documentation

2.67.4.1 Slab_block* Bastei::Slab_block::next

2.67.4.2 Slab_block* Bastei::Slab_block::prev

2.68 Bastei::Slab_entry Class Reference

```
#include <slab.h>
```

Public Member Functions

- void [init](#) ()
- void [occupy](#) (Slab_block *sb)
- void [free](#) ()
- void * [addr](#) ()

Static Public Member Functions

- static [Slab_entry](#) * [slab_entry](#) (void *addr)

2.68.1 Member Function Documentation

2.68.1.1 void Bastei::Slab_entry::init () [inline]

2.68.1.2 void Bastei::Slab_entry::occupy (Slab_block * **sb**) [inline]

2.68.1.3 void Bastei::Slab_entry::free () [inline]

2.68.1.4 void* Bastei::Slab_entry::addr () [inline]

2.68.1.5 static Slab_entry* Bastei::Slab_entry::slab_entry (void * **addr**) [inline, static]

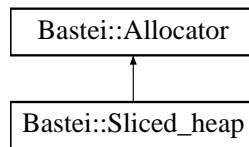
Lookup [Slab_entry](#) by given address

The specified address is supposed to point to `_data[0]`.

2.69 Bastei::Sliced_heap Class Reference

```
#include <heap.h>
```

Inheritance diagram for Bastei::Sliced_heap::



Public Member Functions

- [Sliced_heap](#) ([Ram_session](#) *ram_session, [Rm_session](#) *rm_session)
- [~Sliced_heap](#) ()
- void * [alloc](#) (size_t size)
- void [free](#) (void *addr, size_t size)
- size_t [consumed](#) ()
- size_t [overhead](#) ()

2.69.1 Detailed Description

[Heap](#) that allocates each block at a separate dataspace

2.69.2 Constructor & Destructor Documentation

2.69.2.1 [Bastei::Sliced_heap::Sliced_heap](#) ([Ram_session](#) * *ram_session*, [Rm_session](#) * *rm_session*)

Constructor

2.69.2.2 [Bastei::Sliced_heap::~~Sliced_heap](#) ()

Destructor

2.69.3 Member Function Documentation

2.69.3.1 void* [Bastei::Sliced_heap::alloc](#) (size_t *size*) [virtual]

Allocate block

Implements [Bastei::Allocator](#).

2.69.3.2 void [Bastei::Sliced_heap::free](#) (void * *addr*, size_t *size*) [virtual]

Free block a previously allocated block

Implements [Bastei::Allocator](#).

2.69.3.3 size_t [Bastei::Sliced_heap::consumed](#) () [inline, virtual]

Return total amount of backingstore consumed by the allocator

Reimplemented from [Bastei::Allocator](#).

2.69.3.4 size_t Bastei::Sliced_heap::overhead () [virtual]

Return metadata overhead per block

Implements [Bastei::Allocator](#).

2.70 Bastei::Task Class Reference

```
#include <task.h>
```

Public Member Functions

- [Task](#) ([Capability](#) elf_data_ds_cap, [Capability](#) ram_session_cap, [Capability](#) cpu_session_cap, [Capability](#) parent_cap, const char *name, char *const argv[])
- [~Task](#) ()
- [Capability](#) rm_session_cap () const
- [Capability](#) task_session_cap () const

2.70.1 Constructor & Destructor Documentation**2.70.1.1 Bastei::Task::Task (Capability *elf_data_ds_cap*, Capability *ram_session_cap*, Capability *cpu_session_cap*, Capability *parent_cap*, const char * *name*, char *const *argv*[])**

Constructor - start task

On construction of a task, the execution of the new task is started immediately.

Parameters:

elf_data_ds_cap [Dataspaces](#) that contains the elf binary. This dataspaces can be read-only.

ram_session_cap RAM session providing the BSS for the new task.

cpu_session_cap CPU session for the new task.

parent_cap [Parent](#) of the new task

name Name of task (can be used in debugging)

argv ???

2.70.1.2 Bastei::Task::~~Task ()

Destructor - kill task

2.70.2 Member Function Documentation**2.70.2.1 Capability Bastei::Task::rm_session_cap () const** [inline]**2.70.2.2 Capability Bastei::Task::task_session_cap () const** [inline]

2.71 Bastei::Task_session Class Reference

```
#include <task_session.h>
```

Public Member Functions

- virtual [~Task_session](#) ()
- virtual int [bind_thread](#) ([Capability](#) thread)=0

Protected Types

- enum [Opcode](#) { [BIND_THREAD](#) }

2.71.1 Member Enumeration Documentation

2.71.1.1 enum Bastei::Task_session::Opcode [protected]

Enumerator:

BIND_THREAD

2.71.2 Constructor & Destructor Documentation

2.71.2.1 virtual Bastei::Task_session::~~Task_session () [inline, virtual]

2.71.3 Member Function Documentation

2.71.3.1 virtual int Bastei::Task_session::bind_thread (Capability *thread*) [pure virtual]

Bind thread to task

Parameters:

thread capability of thread to bind

Returns:

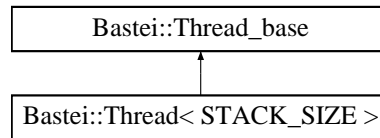
zero on success or negative error code

After successful bind the thread will run inside this task when started.

2.72 Bastei::Thread< STACK_SIZE > Class Template Reference

```
#include <thread.h>
```

Inheritance diagram for Bastei::Thread< STACK_SIZE >::



Public Member Functions

- [Thread](#) (const char *name="<noname>")
- void [start](#) ()

`template<int STACK_SIZE> class Bastei::Thread< STACK_SIZE >`

2.72.1 Constructor & Destructor Documentation

2.72.1.1 `template<int STACK_SIZE> Bastei::Thread< STACK_SIZE >::Thread (const char * name = "<noname>") [inline, explicit]`

Constructor

Parameters:

name [Thread](#) name (for debugging)

2.72.2 Member Function Documentation

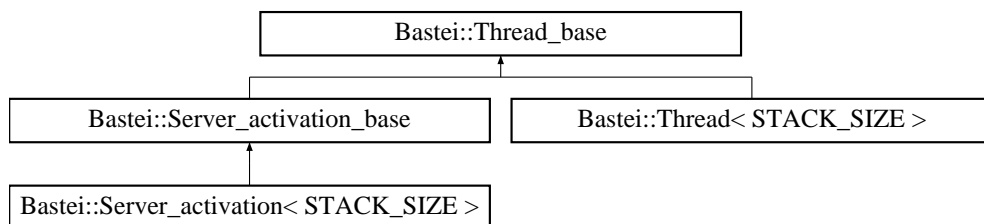
2.72.2.1 `template<int STACK_SIZE> void Bastei::Thread< STACK_SIZE >::start () [inline]`

Start execution of the thread

2.73 Bastei::Thread_base Class Reference

```
#include <thread.h>
```

Inheritance diagram for Bastei::Thread_base::



Public Member Functions

- virtual [~Thread_base](#) ()
- virtual void [entry](#) ()=0

- void [cancel_blocking](#) ()
- Fiasco::l4_threadid_t [tid](#) ()

Protected Member Functions

- void [_start](#) (void *sp, const char *name="<noname>")

Static Protected Member Functions

- static void [_thread_start](#) (void *arg)

Protected Attributes

- [Bastei::Capability_thread_cap](#)
- Fiasco::l4_threadid_t [_tid](#)

Static Protected Attributes

- static [Lock_startup_arg_lock](#)
- static [Thread_base](#) * [_startup_arg](#)

2.73.1 Detailed Description

[Thread_base](#) provides the functionality to start the execution of an activity.

2.73.2 Constructor & Destructor Documentation

2.73.2.1 virtual [Bastei::Thread_base::~~Thread_base](#) () [virtual]

2.73.3 Member Function Documentation

2.73.3.1 void [Bastei::Thread_base::_start](#) (void * *sp*, const char * *name* = "<noname>") [protected]

Start execution of thread at entry function

This function should be called from the constructor of the derived class.

Parameters:

sp Initial stack pointer

name Name of thread (used for debugging)

2.73.3.2 static void [Bastei::Thread_base::_thread_start](#) (void * *arg*) [static, protected]

[Thread](#) startup wrapper

2.73.3.3 virtual void Bastei::Thread_base::entry () [pure virtual]

Entry function of the thread

Implemented in [Bastei::Server_activation_base](#).

2.73.3.4 void Bastei::Thread_base::cancel_blocking ()

Cancel currently blocking operation

2.73.3.5 Fiasco::!4_threadid_t Bastei::Thread_base::tid () [inline]

Only to be called from [Fiasco](#) specific code

2.73.4 Member Data Documentation**2.73.4.1 Bastei::Capability Bastei::Thread_base::_thread_cap** [protected]

Capability for this thread (set by [_start\(\)](#))

2.73.4.2 Fiasco::!4_threadid_t Bastei::Thread_base::_tid [protected]

[Fiasco](#) thread ID that will be used to create capabilities to endpoints managed by the thread.

2.73.4.3 Lock Bastei::Thread_base::_startup_arg_lock [static, protected]

[Thread](#) startup argument frame

2.73.4.4 Thread_base* Bastei::Thread_base::_startup_arg [static, protected]**2.74 Bastei::Thread_state Struct Reference**

```
#include <thread_state.h>
```

Public Member Functions

- [Thread_state \(\)](#)

Public Attributes

- [addr_t ip](#)
- [addr_t sp](#)

2.74.1 Constructor & Destructor Documentation**2.74.1.1 Bastei::Thread_state::Thread_state ()** [inline]

Constructor

2.74.2 Member Data Documentation

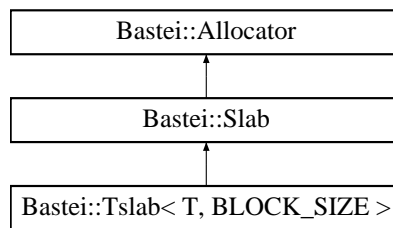
2.74.2.1 addr_t Bastei::Thread_state::ip

2.74.2.2 addr_t Bastei::Thread_state::sp

2.75 Bastei::Tslab< T, BLOCK_SIZE > Class Template Reference

```
#include <tslab.h>
```

Inheritance diagram for Bastei::Tslab< T, BLOCK_SIZE >::



Public Member Functions

- [Tslab](#) ([Allocator](#) *backingstore, [Slab_block](#) *initial_sb=0)
- T * [first_object](#) ()

```
template<typename T, size_t BLOCK_SIZE> class Bastei::Tslab< T, BLOCK_SIZE >
```

2.75.1 Constructor & Destructor Documentation

2.75.1.1 `template<typename T, size_t BLOCK_SIZE> Bastei::Tslab< T, BLOCK_SIZE >::Tslab (Allocator * backingstore, Slab_block * initial_sb = 0) [inline]`

2.75.2 Member Function Documentation

2.75.2.1 `template<typename T, size_t BLOCK_SIZE> T* Bastei::Tslab< T, BLOCK_SIZE >::first_object () [inline]`

2.76 Nitpicker::View Class Reference

```
#include <nitpicker_view.h>
```

Public Member Functions

- virtual [~View](#) ()
- virtual int [viewport](#) (int x, int y, int w, int h, int buf_x, int buf_y, bool redraw=true)=0

- virtual int `stack` (`Bastei::Capability` neighbor=`Bastei::Capability`(), bool behind=true, bool redraw=true)=0
- virtual int `title` (const char *title)=0

Protected Types

- enum `Opcode` { `VIEWPORT`, `STACK`, `TITLE` }

2.76.1 Member Enumeration Documentation

2.76.1.1 enum `Nitpicker::View::Opcode` [protected]

Enumerator:

`VIEWPORT`
`STACK`
`TITLE`

2.76.2 Constructor & Destructor Documentation

2.76.2.1 virtual `Nitpicker::View::~~View` () [inline, virtual]

2.76.3 Member Function Documentation

2.76.3.1 virtual int `Nitpicker::View::viewport` (int *x*, int *y*, int *w*, int *h*, int *buf_x*, int *buf_y*, bool *redraw* = true) [pure virtual]

Define position and viewport

Both attributes are handled in one function to enable atomic updates of position and viewport. This is the common case for moving an overlay window.

2.76.3.2 virtual int `Nitpicker::View::stack` (`Bastei::Capability` *neighbor* = `Bastei::Capability` (), bool *behind* = true, bool *redraw* = true) [pure virtual]

Reposition view in view stack

Parameters:

neighbor Neighbor view
behind Insert view in front (true) or behind (false) the specified neighbor
redraw Redraw affected screen region

To insert a view at the top of the view stack, specify *neighbor* = invalid and *behind* = true. To insert a view at the bottom of the view stack, specify *neighbor* = invalid and *behind* = false.

2.76.3.3 virtual int `Nitpicker::View::title` (const char * *title*) [pure virtual]

Assign new view title

3 Bastei OS Architecture File Documentation

3.1 base/allocator.h File Reference

```
#include <base/stdint.h>
#include <base/exception.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Allocator](#)
- class [Bastei::Allocator::Out_of_memory](#)

Functions

- `template<typename T>`
void [Bastei::destroy](#) (Allocator *alloc, T *obj)
- void * [operator new](#) (Bastei::size_t size, [Bastei::Allocator](#) *allocator)

3.1.1 Function Documentation

3.1.1.1 void* operator new (Bastei::size_t size, Bastei::Allocator * allocator)

3.2 base/allocator_avl.h File Reference

```
#include <base/allocator.h>
#include <base/tslab.h>
#include <util/avl_tree.h>
#include <util/misc_math.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Allocator_avl_base](#)
- class [Bastei::Allocator_avl_base::Block](#)
- class [Bastei::Allocator_avl_tpl< BMDT >](#)
- class [Bastei::Allocator_avl_tpl< BMDT >::Block](#)
- class [Bastei::Empty](#)

Typedefs

- typedef Allocator_avl_tpl< Empty > [Bastei::Allocator_avl](#)

3.3 base/capability.h File Reference

```
#include <l4/sys/types.h>
```

Namespaces

- namespace [Fiasco](#)
- namespace [Bastei](#)

Classes

- class [Fiasco::Capability](#)

Typedefs

- typedef [::Fiasco::Capability](#) [Bastei::Capability](#)
- typedef [::Fiasco::l4_threadid_t](#) [Bastei::Connection_state](#)

3.4 base/child.h File Reference

```
#include <parent/server.h>
#include <ram_session/client.h>
#include <util/arg_string.h>
#include <base/env.h>
#include <base/server.h>
#include <base/heap.h>
#include <base/task.h>
#include <base/service.h>
#include <base/lock.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Session](#)
- class [Bastei::Child](#)

3.5 *init/child.h* File Reference

```
#include <base/env.h>
#include <base/child.h>
#include <base/service.h>
#include <cap_session/cap_session.h>
#include <rom_session/server.h>
#include <base/snprintf.h>
```

Namespaces

- namespace [Init](#)

Classes

- class [Init::Child](#)
- class [Init::Child::Config_rom_session_component](#)

3.6 *base/env.h* File Reference

```
#include <base/capability.h>
#include <parent/parent.h>
#include <rm_session/rm_session.h>
#include <ram_session/ram_session.h>
#include <cpu_session/cpu_session.h>
#include <task_session/task_session.h>
#include <base/allocator.h>
#include <base/snprintf.h>
#include <base/lock.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Env](#)

Functions

- `Env * Bastei::env ()`

- Capability [Bastei::parent_cap](#) ()
- Capability [Bastei::session](#) (const char *service_name, const char *format_args,...)

3.7 base/errno.h File Reference

Namespaces

- namespace [Bastei](#)

Enumerations

- enum { [Bastei::ERR_INVALID_OBJECT](#) = -1 }

3.8 base/heap.h File Reference

```
#include <util/list.h>
#include <ram_session/ram_session.h>
#include <rm_session/rm_session.h>
#include <base/allocator_avl.h>
#include <base/lock.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Heap](#)
- class [Bastei::Heap::Dataspace](#)
- class [Bastei::Heap::Dataspace_pool](#)
- class [Bastei::Sliced_heap](#)

3.9 base/ipc.h File Reference

```
#include <base/ipc_generic.h>
```

3.10 base/ipc_generic.h File Reference

```
#include <util/misc_math.h>
#include <util/string.h>
#include <base/exception.h>
#include <base/capability.h>
#include <base/ipc_msgbuf.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Ipc_error](#)
- class [Bastei::Buffer](#)
- class [Bastei::Ipc_marshall](#)
- class [Bastei::Ipc_unmarshaller](#)
- class [Bastei::Ipc_ostream](#)
- class [Bastei::Ipc_istream](#)
- class [Bastei::Ipc_client](#)
- class [Bastei::Ipc_server](#)

Enumerations

- enum [Bastei::Ipc_ostream_send](#) { [Bastei::IPC_SEND](#) }
- enum [Bastei::Ipc_istream_wait](#) { [Bastei::IPC_WAIT](#) }
- enum [Bastei::Ipc_client_call](#) { [Bastei::IPC_CALL](#) }
- enum [Bastei::Ipc_server_reply_wait](#) { [Bastei::IPC_REPLY_WAIT](#) }

3.11 base/ipc_msgbuf.h File Reference

Namespaces

- namespace [Fiasco](#)
- namespace [Bastei](#)

Classes

- class [Fiasco::Msgbuf_base](#)
- class [Fiasco::Msgbuf< BUF_SIZE >](#)
- class [Bastei::Msgbuf< BUF_SIZE >](#)

Typedefs

- typedef [::Fiasco::Msgbuf_base](#) [Bastei::Msgbuf_base](#)

3.12 base/lock.h File Reference

```
#include <base/cancelable_lock.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Lock](#)

3.13 base/lock_guard.h File Reference

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Lock_guard< LT >](#)

3.14 base/object_pool.h File Reference

```
#include <util/avl_tree.h>
```

```
#include <base/lock.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Object_pool< OBJ_TYPE >](#)
- class [Bastei::Object_pool< OBJ_TYPE >::Entry](#)

3.15 base/ram_dataspace.h File Reference

```
#include <ram_session/ram_session.h>
```

```
#include <base/env.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Ram_dataspace](#)

3.16 *base/semaphore.h* File Reference

```
#include <base/lock.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Semaphore](#)

3.17 *base/server.h* File Reference

```
#include <base/thread.h>
#include <base/errno.h>
#include <base/ipc.h>
#include <base/object_pool.h>
#include <base/lock.h>
#include <cap_session/cap_session.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Server_object](#)
- class [Bastei::Server_activation_base](#)
- class [Bastei::Server_entrypoint](#)
- class [Bastei::Server_activation< STACK_SIZE >](#)

3.18 *base/service.h* File Reference

```
#include <root/client.h>
#include <base/lock.h>
#include <base/printf.h>
#include <util/list.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Client](#)
- class [Bastei::Service](#)
- class [Bastei::Local_service](#)
- class [Bastei::Remote_service](#)
- class [Bastei::Service_pool< ST >](#)
- class [Bastei::Remote_service_pool](#)

Typedefs

- typedef [Service_pool< Local_service > Bastei::Local_service_pool](#)

3.19 base/session_control.h File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Session_control](#)

3.20 base/slab.h File Reference

```
#include <base/allocator.h>
```

```
#include <base/stdint.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Slab_block](#)
- class [Bastei::Slab_entry](#)
- class [Bastei::Slab](#)

3.21 base/sleep.h File Reference

```
#include <base/ipc.h>
```

Namespaces

- namespace [Bastei](#)

Functions

- void [Bastei::sleep_forever](#) ()

3.22 *base/task.h* File Reference

```
#include <base/capability.h>
#include <rm_session/client.h>
#include <task_session/client.h>
#include <cpu_session/client.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Task](#)

3.23 *base/thread.h* File Reference

```
#include <base/capability.h>
#include <base/lock.h>
#include <l4/sys/types.h>
```

Namespaces

- namespace [Fiasco](#)
- namespace [Bastei](#)

Classes

- class [Bastei::Thread_base](#)
- class [Bastei::Thread](#)< [STACK_SIZE](#) >

3.24 *base/thread_state.h* File Reference

```
#include <base/stdint.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- struct [Bastei::Thread_state](#)

3.25 base/tslab.h File Reference

```
#include <base/slab.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Tslab< T, BLOCK_SIZE >](#)

3.26 cap_session/cap_session.h File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Cap_session](#)

3.27 cpu_session/cpu_session.h File Reference

```
#include <base/stdint.h>  
#include <base/capability.h>  
#include <base/thread_state.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Cpu_session](#)

3.28 *dataspace/dataspace.h* File Reference

```
#include <base/stdint.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Dataspace](#)

3.29 *framebuffer_session/framebuffer_session.h* File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Framebuffer](#)

Classes

- class [Framebuffer::Session](#)

3.30 *init/child_config.h* File Reference

```
#include <base/env.h>
```

```
#include <base/printf.h>
```

```
#include <util/xml_node.h>
```

```
#include <rom_session/client.h>
```

```
#include <ram_session/client.h>
```

Namespaces

- namespace [Init](#)

Classes

- class [Init::Child_config](#)

3.31 input_session/input_session.h File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Input](#)

Classes

- class [Input::Session](#)

3.32 io_mem_session/io_mem_session.h File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Io_mem_session](#)

3.33 io_port_session/io_port_session.h File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Io_port_session](#)

3.34 irq_session/irq_session.h File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Irq_session](#)

3.35 *log_session/log_session.h* File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Log_session](#)

3.36 *nitpicker_session/nitpicker_session.h* File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Nitpicker](#)

Classes

- class [Nitpicker::Session](#)

3.37 *nitpicker_view/nitpicker_view.h* File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Nitpicker](#)

Classes

- class [Nitpicker::View](#)

3.38 *parent/parent.h* File Reference

```
#include <base/ipc.h>
```

```
#include <base/session_control.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Parent](#)

3.39 pci_device/pci_device.h File Reference

Namespaces

- namespace [Pci](#)

Classes

- class [Pci::Device](#)
- class [Pci::Device::Resource](#)

3.40 pci_session/pci_session.h File Reference

Namespaces

- namespace [Pci](#)

Classes

- class [Pci::Session](#)

3.41 ram_session/ram_session.h File Reference

```
#include <base/stdint.h>
#include <base/capability.h>
#include <base/exception.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Ram_session](#)
- class [Bastei::Ram_session::Alloc_failed](#)
- class [Bastei::Ram_session::Quota_exceeded](#)

3.42 *rm_session/rm_session.h* File Reference

```
#include <base/exception.h>
#include <base/stdint.h>
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Rm_session](#)
- class [Bastei::Rm_session::Attach_failed](#)
- class [Bastei::Rm_session::Invalid_dataspace](#)
- class [Bastei::Rm_session::Region_conflict](#)

3.43 *rom_session/rom_session.h* File Reference

```
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Rom_session](#)

3.44 *root/root.h* File Reference

```
#include <base/ipc.h>
#include <base/session_control.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Root](#)

3.45 task_session/task_session.h File Reference

```
#include <base/stdint.h>
#include <base/capability.h>
```

Namespaces

- namespace [Bastei](#)

Classes

- class [Bastei::Task_session](#)

3.46 timer_session/timer_session.h File Reference

Namespaces

- namespace [Timer](#)

Classes

- class [Timer::Session](#)

Index

- `_add_session`
 - `Bastei::Child`, 17
- `_argbuf`
 - `Init::Child`, 21
- `_call`
 - `Bastei::Ipc_client`, 36
- `_cap_valid`
 - `Bastei::Server_activation_base`, 68
- `_curr_obj`
 - `Bastei::Server_activation_base`, 68
- `_curr_obj_lock`
 - `Bastei::Server_activation_base`, 68
- `_delay_start`
 - `Bastei::Server_activation_base`, 68
- `_delegate_to_parent`
 - `Init::Child`, 20
- `_dst`
 - `Bastei::Ipc_ostream`, 42
- `_find_by_address`
 - `Bastei::Allocator_avl_base`, 8
- `_local_name`
 - `Fiasco::Capability`, 16
- `_long_at_idx`
 - `Bastei::Ipc_unmarshaller`, 45
- `_msg_start`
 - `Fiasco::Msgbuf_base`, 51
- `_prepare_next_call`
 - `Bastei::Ipc_client`, 36
- `_prepare_next_receive`
 - `Bastei::Ipc_istream`, 38
- `_prepare_next_reply_wait`
 - `Bastei::Ipc_server`, 43
- `_prepare_next_send`
 - `Bastei::Ipc_ostream`, 41
- `_prepend_label`
 - `Init::Child`, 20
- `_ram_quota`
 - `Init::Child`, 20
- `_ram_session_cap`
 - `Bastei::Child`, 19
- `_ram_session_client`
 - `Bastei::Child`, 19
- `_rcv_cs`
 - `Bastei::Ipc_istream`, 39
- `_rcv_msg`
 - `Bastei::Ipc_istream`, 39
- `_rcvbuf`
 - `Bastei::Ipc_unmarshaller`, 45
- `_rcvbuf_size`
 - `Bastei::Ipc_unmarshaller`, 45
- `_read_bytebuf_from_buf`
 - `Bastei::Ipc_unmarshaller`, 45
- `_read_from_buf`
 - `Bastei::Ipc_unmarshaller`, 45
- `_read_offset`
 - `Bastei::Ipc_unmarshaller`, 45
- `_remote_services`
 - `Init::Child`, 21
- `_remove_session`
 - `Bastei::Child`, 17
- `_reply_needed`
 - `Bastei::Ipc_server`, 44
- `_reply_wait`
 - `Bastei::Ipc_server`, 43
- `_result`
 - `Bastei::Ipc_client`, 37
- `_send`
 - `Bastei::Ipc_ostream`, 41
- `_service_wait_queue`
 - `Bastei::Service_pool`, 72
- `_service_wait_queue_lock`
 - `Bastei::Service_pool`, 72
- `_services`
 - `Bastei::Service_pool`, 72
- `_size`
 - `Fiasco::Msgbuf_base`, 51
- `_snd_msg`
 - `Bastei::Ipc_ostream`, 42
- `_sndbuf`
 - `Bastei::Ipc_marshall`, 40
- `_sndbuf_size`
 - `Bastei::Ipc_marshall`, 40
- `_start`
 - `Bastei::Thread_base`, 89
- `_startup_arg`
 - `Bastei::Thread_base`, 90
- `_startup_arg_lock`
 - `Bastei::Thread_base`, 90
- `_thread_cap`
 - `Bastei::Thread_base`, 90
- `_thread_start`
 - `Bastei::Thread_base`, 89
- `_tid`
 - `Bastei::Thread_base`, 90

- Fiasco::Capability, 16
- _wait
 - Bastei::Ipc_istream, 38
- _write_buffer_to_buf
 - Bastei::Ipc_marshalller, 40
- _write_offset
 - Bastei::Ipc_marshalller, 40
- _write_to_buf
 - Bastei::Ipc_marshalller, 40
- ~Allocator
 - Bastei::Allocator, 6
- ~Cap_session
 - Bastei::Cap_session, 14
- ~Child
 - Bastei::Child, 17
 - Init::Child, 20
- ~Child_config
 - Init::Child_config, 21
- ~Client
 - Bastei::Client, 22
- ~Cpu_session
 - Bastei::Cpu_session, 24
- ~Dataspac
 - Bastei::Dataspac, 27
- ~Device
 - Pci::Device, 28
- ~Env
 - Bastei::Env, 30
- ~Io_mem_session
 - Bastei::Io_mem_session, 33
- ~Io_port_session
 - Bastei::Io_port_session, 34
- ~Ipc_istream
 - Bastei::Ipc_istream, 38
- ~Irq_session
 - Bastei::Irq_session, 46
- ~Lock_guard
 - Bastei::Lock_guard, 47
- ~Log_session
 - Bastei::Log_session, 48
- ~Parent
 - Bastei::Parent, 55
- ~Ram_dataspac
 - Bastei::Ram_dataspac, 56
- ~Ram_session
 - Bastei::Ram_session, 58
- ~Rm_session
 - Bastei::Rm_session, 61
- ~Rom_session
 - Bastei::Rom_session, 63
- ~Root
 - Bastei::Root, 64
- ~Server_object
 - Bastei::Server_object, 69
- ~Service
 - Bastei::Service, 70
- ~Session
 - Framebuffer::Session, 76
 - Input::Session, 75
 - Nitpicker::Session, 73
 - Pci::Session, 74
 - Timer::Session, 79
- ~Session_control
 - Bastei::Session_control, 79
- ~Slab
 - Bastei::Slab, 81
- ~Sliced_heap
 - Bastei::Sliced_heap, 85
- ~Task
 - Bastei::Task, 86
- ~Task_session
 - Bastei::Task_session, 87
- ~Thread_base
 - Bastei::Thread_base, 89
- ~View
 - Nitpicker::View, 92
- add_activation
 - Bastei::Server_entrypoint, 69
- add_area
 - Bastei::Allocator_avl_base, 8
 - Bastei::Allocator_avl_tpl, 13
- ADD_CLIENT
 - Bastei::Rm_session, 61
- add_client
 - Bastei::Rm_session, 62
- addr
 - Bastei::Allocator_avl_base::Block, 11
 - Bastei::Buffer, 13
 - Bastei::Slab_entry, 84
 - Fiasco::Msgbuf_base, 50
- ALLOC
 - Bastei::Cap_session, 14
 - Bastei::Ram_session, 57
- alloc
 - Bastei::Allocator, 6
 - Bastei::Allocator_avl_base, 9
 - Bastei::Cap_session, 14
 - Bastei::Heap, 32
 - Bastei::Ram_session, 58

- Bastei::Slab, [81](#), [82](#)
- Bastei::Slab_block, [83](#)
- Bastei::Sliced_heap, [85](#)
- alloc_addr
 - Bastei::Allocator_avl_base, [8](#)
- alloc_aligned
 - Bastei::Allocator_avl_base, [8](#)
- allocator.h
 - operator new, [93](#)
- Allocator_avl
 - Bastei, [3](#)
- Allocator_avl_base
 - Bastei::Allocator_avl_base, [8](#)
- Allocator_avl_tpl
 - Bastei::Allocator_avl_tpl, [12](#)
- ANNOUNCE
 - Bastei::Parent, [55](#)
- announce
 - Bastei::Child, [18](#)
 - Bastei::Parent, [55](#)
 - Init::Child, [21](#)
- any_block_addr
 - Bastei::Allocator_avl_base, [9](#)
- apply_for
 - Bastei::Client, [23](#)
- ATTACH
 - Bastei::Rm_session, [61](#)
- attach
 - Bastei::Rm_session, [62](#)
- avail
 - Bastei::Allocator_avl_base, [9](#)
 - Bastei::Allocator_avl_base::Block, [11](#)
 - Bastei::Ram_session, [59](#)
 - Bastei::Slab_block, [83](#)
- avail_in_subtree
 - Bastei::Allocator_avl_base::Block, [11](#)
- Avl_node< Entry >
 - Bastei::Object_pool::Entry, [54](#)
- BACKGROUND
 - Nitpicker::Session, [73](#)
- background
 - Nitpicker::Session, [73](#)
- backingstore
 - Bastei::Slab, [82](#)
- badge
 - Bastei::Ipc_istream, [38](#)
- base
 - Pci::Device::Resource, [29](#)
- base/allocator.h, [93](#)
- base/allocator_avl.h, [93](#)
- base/capability.h, [94](#)
- base/child.h, [94](#)
- base/env.h, [95](#)
- base/errno.h, [96](#)
- base/heap.h, [96](#)
- base/ipc.h, [96](#)
- base/ipc_generic.h, [96](#)
- base/ipc_msgbuf.h, [97](#)
- base/lock.h, [97](#)
- base/lock_guard.h, [98](#)
- base/object_pool.h, [98](#)
- base/ram_dataspace.h, [98](#)
- base/semaphore.h, [99](#)
- base/server.h, [99](#)
- base/service.h, [99](#)
- base/session_control.h, [100](#)
- base/slab.h, [100](#)
- base/sleep.h, [100](#)
- base/task.h, [101](#)
- base/thread.h, [101](#)
- base/thread_state.h, [101](#)
- base/tslab.h, [102](#)
- base_class
 - Pci::Device, [28](#)
- Bastei, [1](#)
 - Allocator_avl, [3](#)
 - Capability, [3](#)
 - Connection_state, [3](#)
 - destroy, [4](#)
 - env, [4](#)
 - ERR_INVALID_OBJECT, [4](#)
 - IPC_CALL, [4](#)
 - Ipc_client_call, [4](#)
 - Ipc_istream_wait, [4](#)
 - Ipc_ostream_send, [4](#)
 - IPC_REPLY_WAIT, [4](#)
 - IPC_SEND, [4](#)
 - Ipc_server_reply_wait, [4](#)
 - IPC_WAIT, [4](#)
 - Local_service_pool, [3](#)
 - Msgbuf_base, [3](#)
 - parent_cap, [4](#)
 - session, [4](#)
 - sleep_forever, [4](#)
- Bastei::Allocator, [6](#)
 - ~Allocator, [6](#)
 - alloc, [6](#)
 - consumed, [7](#)
 - free, [6](#)

- overhead, 7
- Bastei::Allocator::Out_of_memory, 7
- Bastei::Allocator_avl_base, 7
 - _find_by_address, 8
 - add_area, 8
 - alloc, 9
 - alloc_addr, 8
 - alloc_aligned, 8
 - Allocator_avl_base, 8
 - any_block_addr, 9
 - avail, 9
 - dump_addr_tree, 9
 - free, 9
 - overhead, 9
 - remove_area, 8
- Bastei::Allocator_avl_base::Block
 - FREE, 10
 - USED, 10
- Bastei::Allocator_avl_base::Block, 10
 - addr, 11
 - avail, 11
 - avail_in_subtree, 11
 - Block, 10
 - dump, 11
 - dump_dot, 11
 - find_best_fit, 11
 - find_by_address, 11
 - higher, 11
 - id, 11
 - max_avail, 11
 - recompute, 11
 - size, 11
 - used, 11
- Bastei::Allocator_avl_tpl, 12
 - add_area, 13
 - Allocator_avl_tpl, 12
 - metadata, 13
- Bastei::Buffer, 13
 - addr, 13
 - Buffer, 13
 - size, 13
- Bastei::Cap_session
 - ALLOC, 14
 - FREE, 14
- Bastei::Cap_session, 14
 - ~Cap_session, 14
 - alloc, 14
 - free, 14
 - Opcode, 14
- Bastei::Child, 16
 - _add_session, 17
 - _ram_session_cap, 19
 - _ram_session_client, 19
 - _remove_session, 17
 - ~Child, 17
 - announce, 18
 - Child, 17
 - close, 18
 - cpu_session_cap, 17
 - exit, 18
 - finalize_construction, 17
 - heap, 17
 - parent_entrypoint, 18
 - prepare_destruction, 18
 - ram_session_cap, 17
 - revoke_server, 18
 - session, 18
 - transfer_quota, 18
- Bastei::Client, 22
 - ~Client, 22
 - apply_for, 23
 - Client, 22
 - name, 23
 - sleep, 23
 - wakeup, 23
- Bastei::Cpu_session
 - CANCEL_BLOCKING, 24
 - CREATE_THREAD, 24
 - FIRST, 24
 - KILL_THREAD, 24
 - NAME, 24
 - NEXT, 24
 - SET_PAGER, 24
 - START, 24
 - STATE, 24
 - THREAD_NAME_LEN, 24
- Bastei::Cpu_session, 23
 - ~Cpu_session, 24
 - cancel_blocking, 25
 - create_thread, 24
 - first, 25
 - kill_thread, 24
 - name, 25
 - next, 25
 - Opcode, 24
 - set_pager, 25
 - start, 25
 - state, 26
- Bastei::Dataspace, 26
 - ~Dataspace, 27

- NUM_GENERIC_OPCODES, 26
- Opcode, 26
- PHYS_ADDR, 26
- phys_addr, 27
- SIZE, 26
- size, 27
- Bastei::Empty, 30
- Bastei::Env, 30
 - ~Env, 30
 - cpu_session, 30
 - heap, 31
 - parent, 30
 - ram_session, 30
 - ram_session_cap, 30
 - rm_session, 31
 - task_session, 31
- Bastei::Heap, 31
 - alloc, 32
 - consumed, 32
 - free, 32
 - Heap, 32
 - overhead, 32
 - quota_limit, 32
 - UNLIMITED, 32
- Bastei::Io_mem_session
 - DATASPACE, 33
- Bastei::Io_mem_session, 32
 - ~Io_mem_session, 33
 - dataspace, 33
 - Opcode, 33
- Bastei::Io_port_session
 - INB, 34
 - INL, 34
 - INW, 34
 - OUTB, 34
 - OUTL, 34
 - OUTW, 34
- Bastei::Io_port_session, 33
 - ~Io_port_session, 34
 - inb, 34
 - inl, 34
 - inw, 34
 - Opcode, 34
 - outb, 35
 - outl, 35
 - outw, 35
- Bastei::Ipc_client, 35
 - _call, 36
 - _prepare_next_call, 36
 - _result, 37
 - Ipc_client, 36
 - operator int, 37
 - operator<<, 36, 37
 - operator>>, 37
- Bastei::Ipc_error, 37
- Bastei::Ipc_istream, 37
 - _prepare_next_receive, 38
 - _rcv_cs, 39
 - _rcv_msg, 39
 - _wait, 38
 - ~Ipc_istream, 38
 - badge, 38
 - Ipc_istream, 38
 - operator>>, 38, 39
- Bastei::Ipc_marshall, 39
 - _sndbuf, 40
 - _sndbuf_size, 40
 - _write_buffer_to_buf, 40
 - _write_offset, 40
 - _write_to_buf, 40
 - Ipc_marshall, 40
- Bastei::Ipc_ostream, 40
 - _dst, 42
 - _prepare_next_send, 41
 - _send, 41
 - _snd_msg, 42
 - Ipc_ostream, 41
 - operator<<, 42
 - ready_for_send, 41
- Bastei::Ipc_server, 42
 - _prepare_next_reply_wait, 43
 - _reply_needed, 44
 - _reply_wait, 43
 - Ipc_server, 43
 - operator<<, 43
 - operator>>, 43
 - ret, 43
- Bastei::Ipc_unmarshaller, 44
 - _long_at_idx, 45
 - _rcvbuf, 45
 - _rcvbuf_size, 45
 - _read_bytebuf_from_buf, 45
 - _read_from_buf, 45
 - _read_offset, 45
 - Ipc_unmarshaller, 44
- Bastei::Irq_session
 - WAIT_FOR_IRQ, 45
- Bastei::Irq_session, 45
 - ~Irq_session, 46
 - Opcode, 45

- wait_for_irq, 46
- Bastei::Local_service, 46
 - Local_service, 46
 - root, 46
- Bastei::Lock, 46
 - Guard, 47
 - Lock, 47
 - lock, 47
- Bastei::Lock_guard, 47
 - ~Lock_guard, 47
 - Lock_guard, 47
- Bastei::Log_session
 - WRITE, 48
- Bastei::Log_session, 48
 - ~Log_session, 48
 - Opcode, 48
 - write, 48
- Bastei::Msgbuf, 49
- Bastei::Object_pool, 51
 - first, 52
 - insert, 52
 - obj_by_cap, 52
 - obj_by_id, 52
 - remove, 52
- Bastei::Object_pool::Entry
 - OBJ_ID_INVALID, 53
- Bastei::Object_pool::Entry, 52
 - Avl_node< Entry >, 54
 - cap, 53, 54
 - Entry, 53
 - find_by_obj_id, 53
 - higher, 53
 - Object_pool, 54
- Bastei::Parent, 54
 - ~Parent, 55
 - ANNOUNCE, 55
 - announce, 55
 - CLOSE, 55
 - EXIT, 55
 - exit, 55
 - MSGBUF_SIZE, 55
 - Opcode, 55
 - SESSION, 55
 - session, 55
 - TRANSFER_QUOTA, 55
 - transfer_quota, 55
- Bastei::Ram_dataspace, 56
 - ~Ram_dataspace, 56
 - cap, 56
 - local_addr, 56
 - Ram_dataspace, 56
- Bastei::Ram_session
 - ALLOC, 57
 - FREE, 57
 - QUOTA, 57
 - REF_ACCOUNT, 57
 - TRANSFER_QUOTA, 57
 - USED, 57
- Bastei::Ram_session, 57
 - ~Ram_session, 58
 - alloc, 58
 - avail, 59
 - free, 58
 - Opcode, 57
 - quota, 59
 - ref_account, 58
 - transfer_quota, 58
 - used, 59
- Bastei::Ram_session::Alloc_failed, 59
- Bastei::Ram_session::Quota_exceeded, 59
- Bastei::Remote_service, 59
 - Remote_service, 60
 - root, 60
 - server, 60
- Bastei::Remote_service_pool, 60
 - find_by_server, 61
- Bastei::Rm_session
 - ADD_CLIENT, 61
 - ATTACH, 61
 - DETACH, 61
- Bastei::Rm_session, 61
 - ~Rm_session, 61
 - add_client, 62
 - attach, 62
 - detach, 62
 - Opcode, 61
- Bastei::Rm_session::Attach_failed, 62
- Bastei::Rm_session::Invalid_dataspace, 63
- Bastei::Rm_session::Region_conflict, 63
- Bastei::Rom_session
 - DATASPACE, 63
- Bastei::Rom_session, 63
 - ~Rom_session, 63
 - dataspace, 64
 - Opcode, 63
- Bastei::Root, 64
 - ~Root, 64
 - CLOSE, 64
 - Opcode, 64
 - SESSION, 64

- session, 64
- Bastei::Semaphore, 65
 - down, 65
 - Semaphore, 65
 - up, 65
- Bastei::Server_activation, 65
 - Server_activation, 66
 - start, 66
- Bastei::Server_activation_base, 66
 - _cap_valid, 68
 - _curr_obj, 68
 - _curr_obj_lock, 68
 - _delay_start, 68
 - cap, 67
 - entry, 67
 - ep, 67
 - leave_server_object, 67
 - Server_activation_base, 67
- Bastei::Server_entrpoint, 68
 - add_activation, 69
 - dissolve, 69
 - manage, 69
 - Server_entrpoint, 68
- Bastei::Server_object, 69
 - ~Server_object, 69
 - dispatch, 70
 - lock, 69
 - unlock, 69
- Bastei::Service, 70
 - ~Service, 70
 - name, 70
 - root, 70
 - Service, 70
- Bastei::Service_pool, 71
 - _service_wait_queue, 72
 - _service_wait_queue_lock, 72
 - _services, 72
 - dump_service_wait_queue, 71
 - find, 71
 - insert, 72
 - remove, 72
 - wait_for_service, 71
- Bastei::Session, 77
 - donated_ram_quota, 78
 - ident, 78
 - server, 78
 - Session, 78
 - session_cap, 78
 - session_control, 78
 - valid, 78
- Bastei::Session_control, 79
 - ~Session_control, 79
 - close, 79
- Bastei::Slab, 80
 - ~Slab, 81
 - alloc, 81, 82
 - backingstore, 82
 - block_size, 81
 - consumed, 82
 - dump_sb_list, 81
 - entry_size, 81
 - first_used_elem, 81
 - free, 81, 82
 - insert_sb, 81
 - num_elem, 81
 - num_free_entries_higher_than, 81
 - overhead, 82
 - remove_sb, 81
 - Slab, 81
 - slab_size, 81
- Bastei::Slab_block, 82
 - alloc, 83
 - avail, 83
 - check_bounds, 83
 - dec_avail, 83
 - dump, 83
 - first_used_entry, 83
 - inc_avail, 83
 - next, 84
 - prev, 84
 - slab, 83
 - Slab_block, 83
- Bastei::Slab_entry, 84
 - addr, 84
 - free, 84
 - init, 84
 - occupy, 84
 - slab_entry, 84
- Bastei::Sliced_heap, 84
 - ~Sliced_heap, 85
 - alloc, 85
 - consumed, 85
 - free, 85
 - overhead, 85
 - Sliced_heap, 85
- Bastei::Task, 86
 - ~Task, 86
 - rm_session_cap, 86
 - Task, 86
 - task_session_cap, 86

- Bastei::Task_session
 - BIND_THREAD, 87
- Bastei::Task_session, 87
 - ~Task_session, 87
 - bind_thread, 87
 - Opcode, 87
- Bastei::Thread, 87
 - start, 88
 - Thread, 88
- Bastei::Thread_base, 88
 - _start, 89
 - _startup_arg, 90
 - _startup_arg_lock, 90
 - _thread_cap, 90
 - _thread_start, 89
 - _tid, 90
 - ~Thread_base, 89
 - cancel_blocking, 90
 - entry, 89
 - tid, 90
- Bastei::Thread_state, 90
 - ip, 91
 - sp, 91
 - Thread_state, 90
- Bastei::Tslab, 91
 - first_object, 91
 - Tslab, 91
- BIND_THREAD
 - Bastei::Task_session, 87
- bind_thread
 - Bastei::Task_session, 87
- Block
 - Bastei::Allocator_avl_base::Block, 10
- block_size
 - Bastei::Slab, 81
- buf
 - Fiasco::Msgbuf, 49
 - Fiasco::Msgbuf_base, 51
- Buffer
 - Bastei::Buffer, 13
- CANCEL_BLOCKING
 - Bastei::Cpu_session, 24
- cancel_blocking
 - Bastei::Cpu_session, 25
 - Bastei::Thread_base, 90
- cap
 - Bastei::Object_pool::Entry, 53, 54
 - Bastei::Ram_dataspace, 56
 - Bastei::Server_activation_base, 67
 - cap_session/cap_session.h, 102
- Capability
 - Bastei, 3
 - Fiasco::Capability, 15
- check_bounds
 - Bastei::Slab_block, 83
- Child
 - Bastei::Child, 17
 - Init::Child, 20
- Child_config
 - Init::Child_config, 21
- CLASS_CODE
 - Pci::Device, 27
- class_code
 - Pci::Device, 28
- Client
 - Bastei::Client, 22
- CLOSE
 - Bastei::Parent, 55
 - Bastei::Root, 64
- close
 - Bastei::Child, 18
 - Bastei::Session_control, 79
- Connection_state
 - Bastei, 3
- consumed
 - Bastei::Allocator, 7
 - Bastei::Heap, 32
 - Bastei::Slab, 82
 - Bastei::Sliced_heap, 85
- cpu_session
 - Bastei::Env, 30
- cpu_session/cpu_session.h, 102
- cpu_session_cap
 - Bastei::Child, 17
- CREATE_THREAD
 - Bastei::Cpu_session, 24
- create_thread
 - Bastei::Cpu_session, 24
- CREATE_VIEW
 - Nitpicker::Session, 73
- create_view
 - Nitpicker::Session, 73
- DATASPACE
 - Bastei::Io_mem_session, 33
 - Bastei::Rom_session, 63
 - Framebuffer::Session, 76
 - Input::Session, 75
- dataspace

- Bastei::Io_mem_session, 33
- Bastei::Rom_session, 64
- Framebuffer::Session, 77
- Init::Child_config, 22
- Input::Session, 75
- dataspace/dataspace.h, 103
- dec_avail
 - Bastei::Slab_block, 83
- destroy
 - Bastei, 4
- DESTROY_VIEW
 - Nitpicker::Session, 73
- destroy_view
 - Nitpicker::Session, 73
- DETACH
 - Bastei::Rm_session, 61
- detach
 - Bastei::Rm_session, 62
- DEVICE_ID
 - Pci::Device, 27
- device_id
 - Pci::Device, 28
- dispatch
 - Bastei::Server_object, 70
- dissolve
 - Bastei::Server_entrypoint, 69
- donated_ram_quota
 - Bastei::Session, 78
- down
 - Bastei::Semaphore, 65
- dump
 - Bastei::Allocator_avl_base::Block, 11
 - Bastei::Slab_block, 83
- dump_addr_tree
 - Bastei::Allocator_avl_base, 9
- dump_dot
 - Bastei::Allocator_avl_base::Block, 11
- dump_sb_list
 - Bastei::Slab, 81
- dump_service_wait_queue
 - Bastei::Service_pool, 71
- Entry
 - Bastei::Object_pool::Entry, 53
- entry
 - Bastei::Server_activation_base, 67
 - Bastei::Thread_base, 89
- entry_size
 - Bastei::Slab, 81
- env
 - Bastei, 4
- ep
 - Bastei::Server_activation_base, 67
- ERR_INVALID_OBJECT
 - Bastei, 4
- EXIT
 - Bastei::Parent, 55
- exit
 - Bastei::Child, 18
 - Bastei::Parent, 55
- Fiasco, 5
- Fiasco::Capability, 15
 - _local_name, 16
 - _tid, 16
 - Capability, 15
 - local_name, 15
 - tid, 15
 - valid, 15
- Fiasco::Msgbuf, 48
 - buf, 49
 - Msgbuf, 49
- Fiasco::Msgbuf_base, 50
 - _msg_start, 51
 - _size, 51
 - addr, 50
 - buf, 51
 - rcv_fpage, 51
 - send_dope, 51
 - size, 50
 - size_dope, 51
- finalize_construction
 - Bastei::Child, 17
- find
 - Bastei::Service_pool, 71
- find_best_fit
 - Bastei::Allocator_avl_base::Block, 11
- find_by_address
 - Bastei::Allocator_avl_base::Block, 11
- find_by_obj_id
 - Bastei::Object_pool::Entry, 53
- find_by_server
 - Bastei::Remote_service_pool, 61
- FIND_DEVICE_BY_VENDOR
 - Pci::Session, 74
- find_device_by_vendor
 - Pci::Session, 74
- FIRST
 - Bastei::Cpu_session, 24
- first

- Bastei::Cpu_session, 25
- Bastei::Object_pool, 52
- first_object
 - Bastei::Tslab, 91
- first_used_elem
 - Bastei::Slab, 81
- first_used_entry
 - Bastei::Slab_block, 83
- FLUSH
 - Input::Session, 75
- flush
 - Input::Session, 75
- Framebuffer, 5
- Framebuffer::Session, 76
 - ~Session, 76
 - DATASPACE, 76
 - dataspace, 77
 - INFO, 76
 - info, 77
 - INVALID, 76
 - Mode, 76
 - Opcode, 76
 - REFRESH, 76
 - refresh, 77
 - RGB565, 76
- FRAMEBUFFER_SESSION
 - Nitpicker::Session, 73
- framebuffer_session
 - Nitpicker::Session, 73
- framebuffer_session/framebuffer_session.h, 103
- FREE
 - Bastei::Allocator_avl_base::Block, 10
 - Bastei::Cap_session, 14
 - Bastei::Ram_session, 57
- free
 - Bastei::Allocator, 6
 - Bastei::Allocator_avl_base, 9
 - Bastei::Cap_session, 14
 - Bastei::Heap, 32
 - Bastei::Ram_session, 58
 - Bastei::Slab, 81, 82
 - Bastei::Slab_entry, 84
 - Bastei::Sliced_heap, 85
- Guard
 - Bastei::Lock, 47
- Heap
 - Bastei::Heap, 32
- heap
 - Bastei::Child, 17
 - Bastei::Env, 31
- higher
 - Bastei::Allocator_avl_base::Block, 11
 - Bastei::Object_pool::Entry, 53
- id
 - Bastei::Allocator_avl_base::Block, 11
- ident
 - Bastei::Session, 78
- INB
 - Bastei::Io_port_session, 34
- inb
 - Bastei::Io_port_session, 34
- inc_avail
 - Bastei::Slab_block, 83
- INFO
 - Framebuffer::Session, 76
- info
 - Framebuffer::Session, 77
- Init, 5
- init
 - Bastei::Slab_entry, 84
- init/child.h, 95
- init/child_config.h, 103
- Init::Child, 19
 - _argbuf, 21
 - _delegate_to_parent, 20
 - _prepend_label, 20
 - _ram_quota, 20
 - _remote_services, 21
 - ~Child, 20
 - announce, 21
 - Child, 20
 - name, 20
 - session, 21
- Init::Child_config, 21
 - ~Child_config, 21
 - Child_config, 21
 - dataspace, 22
- INL
 - Bastei::Io_port_session, 34
- inl
 - Bastei::Io_port_session, 34
- Input, 5
- Input::Session, 75
 - ~Session, 75
 - DATASPACE, 75
 - dataspace, 75

- FLUSH, [75](#)
- flush, [75](#)
- IS_PENDING, [75](#)
- is_pending, [75](#)
- Opcode, [75](#)
- INPUT_SESSION
 - Nitpicker::Session, [73](#)
- input_session
 - Nitpicker::Session, [73](#)
- input_session/input_session.h, [104](#)
- insert
 - Bastei::Object_pool, [52](#)
 - Bastei::Service_pool, [72](#)
- insert_sb
 - Bastei::Slab, [81](#)
- INVALID
 - Framebuffer::Session, [76](#)
 - Pci::Device::Resource, [29](#)
- INW
 - Bastei::Io_port_session, [34](#)
- inw
 - Bastei::Io_port_session, [34](#)
- IO
 - Pci::Device::Resource, [29](#)
- io_mem_session/io_mem_session.h, [104](#)
- io_port_session/io_port_session.h, [104](#)
- ip
 - Bastei::Thread_state, [91](#)
- IPC_CALL
 - Bastei, [4](#)
- Ipclient
 - Bastei::Ipclient, [36](#)
- Ipclient_call
 - Bastei, [4](#)
- Ipclient_istream
 - Bastei::Ipclient_istream, [38](#)
- Ipclient_istream_wait
 - Bastei, [4](#)
- Ipclient_marshalller
 - Bastei::Ipclient_marshalller, [40](#)
- Ipclient_ostream
 - Bastei::Ipclient_ostream, [41](#)
- Ipclient_ostream_send
 - Bastei, [4](#)
- IPC_REPLY_WAIT
 - Bastei, [4](#)
- IPC_SEND
 - Bastei, [4](#)
- Ipclient_server
 - Bastei::Ipclient_server, [43](#)
- Ipclient_server_reply_wait
 - Bastei, [4](#)
- Ipclient_unmarshalller
 - Bastei::Ipclient_unmarshalller, [44](#)
- IPC_WAIT
 - Bastei, [4](#)
- irq_session/irq_session.h, [104](#)
- IS_PENDING
 - Input::Session, [75](#)
- is_pending
 - Input::Session, [75](#)
- KILL_THREAD
 - Bastei::Cpu_session, [24](#)
- kill_thread
 - Bastei::Cpu_session, [24](#)
- leave_server_object
 - Bastei::Server_activation_base, [67](#)
- local_addr
 - Bastei::Ram_dataspace, [56](#)
- local_name
 - Fiasco::Capability, [15](#)
- Local_service
 - Bastei::Local_service, [46](#)
- Local_service_pool
 - Bastei, [3](#)
- Lock
 - Bastei::Lock, [47](#)
- lock
 - Bastei::Lock, [47](#)
 - Bastei::Server_object, [69](#)
- Lock_guard
 - Bastei::Lock_guard, [47](#)
- log_session/log_session.h, [105](#)
- manage
 - Bastei::Server_entrypoint, [69](#)
- max_avail
 - Bastei::Allocator_avl_base::Block, [11](#)
- MEMORY
 - Pci::Device::Resource, [29](#)
- metadata
 - Bastei::Allocator_avl_tpl, [13](#)
- Mode
 - Framebuffer::Session, [76](#)
- Msgbuf
 - Fiasco::Msgbuf, [49](#)
- Msgbuf_base
 - Bastei, [3](#)
- MSGBUF_SIZE

- Bastei::Parent, 55
- MSLEEP
 - Timer::Session, 79
- msleep
 - Timer::Session, 79
- NAME
 - Bastei::Cpu_session, 24
- name
 - Bastei::Client, 23
 - Bastei::Cpu_session, 25
 - Bastei::Service, 70
 - Init::Child, 20
- NEXT
 - Bastei::Cpu_session, 24
- next
 - Bastei::Cpu_session, 25
 - Bastei::Slab_block, 84
- Nitpicker, 5
- Nitpicker::Session, 72
 - ~Session, 73
 - BACKGROUND, 73
 - background, 73
 - CREATE_VIEW, 73
 - create_view, 73
 - DESTROY_VIEW, 73
 - destroy_view, 73
 - FRAMEBUFFER_SESSION, 73
 - framebuffer_session, 73
 - INPUT_SESSION, 73
 - input_session, 73
 - Opcode, 73
- Nitpicker::View, 91
 - ~View, 92
 - Opcode, 92
 - STACK, 92
 - stack, 92
 - TITLE, 92
 - title, 92
 - VIEWPORT, 92
 - viewport, 92
- nitpicker_session/nitpicker_session.h, 105
- nitpicker_view/nitpicker_view.h, 105
- num_elem
 - Bastei::Slab, 81
- num_free_entries_higher_than
 - Bastei::Slab, 81
- NUM_GENERIC_OPCODES
 - Bastei::Dataspace, 26
- NUM_RESOURCES
 - Pci::Device, 28
- obj_by_cap
 - Bastei::Object_pool, 52
- obj_by_id
 - Bastei::Object_pool, 52
- OBJ_ID_INVALID
 - Bastei::Object_pool::Entry, 53
- Object_pool
 - Bastei::Object_pool::Entry, 54
- occupy
 - Bastei::Slab_entry, 84
- Opcode
 - Bastei::Cap_session, 14
 - Bastei::Cpu_session, 24
 - Bastei::Dataspace, 26
 - Bastei::Io_mem_session, 33
 - Bastei::Io_port_session, 34
 - Bastei::Irq_session, 45
 - Bastei::Log_session, 48
 - Bastei::Parent, 55
 - Bastei::Ram_session, 57
 - Bastei::Rm_session, 61
 - Bastei::Rom_session, 63
 - Bastei::Root, 64
 - Bastei::Task_session, 87
 - Framebuffer::Session, 76
 - Input::Session, 75
 - Nitpicker::Session, 73
 - Nitpicker::View, 92
 - Pci::Device, 27
 - Pci::Session, 74
 - Timer::Session, 79
- operator int
 - Bastei::Ipc_client, 37
- operator new
 - allocator.h, 93
- operator<<
 - Bastei::Ipc_client, 36, 37
 - Bastei::Ipc_ostream, 42
 - Bastei::Ipc_server, 43
- operator>>
 - Bastei::Ipc_client, 37
 - Bastei::Ipc_istream, 38, 39
 - Bastei::Ipc_server, 43
- OUTB
 - Bastei::Io_port_session, 34
- outb
 - Bastei::Io_port_session, 35
- OUTL

- Bastei::Io_port_session, 34
- outl
 - Bastei::Io_port_session, 35
- OUTW
 - Bastei::Io_port_session, 34
- outw
 - Bastei::Io_port_session, 35
- overhead
 - Bastei::Allocator, 7
 - Bastei::Allocator_avl_base, 9
 - Bastei::Heap, 32
 - Bastei::Slab, 82
 - Bastei::Sliced_heap, 85
- parent
 - Bastei::Env, 30
- parent/parent.h, 105
- parent_cap
 - Bastei, 4
- parent_entrpoint
 - Bastei::Child, 18
- Pci, 5
- Pci::Device, 27
 - ~Device, 28
 - base_class, 28
 - CLASS_CODE, 27
 - class_code, 28
 - DEVICE_ID, 27
 - device_id, 28
 - NUM_RESOURCES, 28
 - Opcode, 27
 - RESOURCE, 27
 - resource, 28
 - sub_class, 28
- Pci::Device::Resource, 28
 - base, 29
 - INVALID, 29
 - IO, 29
 - MEMORY, 29
 - prefetchable, 29
 - Resource, 29
 - size, 29
 - Type, 29
 - type, 29
- Pci::Session, 74
 - ~Session, 74
 - FIND_DEVICE_BY_VENDOR, 74
 - find_device_by_vendor, 74
 - Opcode, 74
 - RELEASE_DEVICE, 74
 - release_device, 74
 - pci_device/pci_device.h, 106
 - pci_session/pci_session.h, 106
 - PHYS_ADDR
 - Bastei::Dataspace, 26
 - phys_addr
 - Bastei::Dataspace, 27
 - prefetchable
 - Pci::Device::Resource, 29
 - prepare_destruction
 - Bastei::Child, 18
 - prev
 - Bastei::Slab_block, 84
- QUOTA
 - Bastei::Ram_session, 57
- quota
 - Bastei::Ram_session, 59
- quota_limit
 - Bastei::Heap, 32
- Ram_dataspace
 - Bastei::Ram_dataspace, 56
- ram_session
 - Bastei::Env, 30
- ram_session/ram_session.h, 106
- ram_session_cap
 - Bastei::Child, 17
 - Bastei::Env, 30
- rcv_fpage
 - Fiasco::Msgbuf_base, 51
- ready_for_send
 - Bastei::Ipc_ostream, 41
- recompute
 - Bastei::Allocator_avl_base::Block, 11
- REF_ACCOUNT
 - Bastei::Ram_session, 57
- ref_account
 - Bastei::Ram_session, 58
- REFRESH
 - Framebuffer::Session, 76
- refresh
 - Framebuffer::Session, 77
- RELEASE_DEVICE
 - Pci::Session, 74
- release_device
 - Pci::Session, 74
- Remote_service
 - Bastei::Remote_service, 60
- remove

- Bastei::Object_pool, 52
- Bastei::Service_pool, 72
- remove_area
 - Bastei::Allocator_avl_base, 8
- remove_sb
 - Bastei::Slab, 81
- RESOURCE
 - Pci::Device, 27
- Resource
 - Pci::Device::Resource, 29
- resource
 - Pci::Device, 28
- ret
 - Bastei::Ipc_server, 43
- revoke_server
 - Bastei::Child, 18
- RGB565
 - Framebuffer::Session, 76
- rm_session
 - Bastei::Env, 31
- rm_session/rm_session.h, 107
- rm_session_cap
 - Bastei::Task, 86
- rom_session/rom_session.h, 107
- root
 - Bastei::Local_service, 46
 - Bastei::Remote_service, 60
 - Bastei::Service, 70
- root/root.h, 107
- Semaphore
 - Bastei::Semaphore, 65
- send_dope
 - Fiasco::Msgbuf_base, 51
- server
 - Bastei::Remote_service, 60
 - Bastei::Session, 78
- Server_activation
 - Bastei::Server_activation, 66
- Server_activation_base
 - Bastei::Server_activation_base, 67
- Server_entrpoint
 - Bastei::Server_entrpoint, 68
- Service
 - Bastei::Service, 70
- SESSION
 - Bastei::Parent, 55
 - Bastei::Root, 64
- Session
 - Bastei::Session, 78
- session
 - Bastei, 4
 - Bastei::Child, 18
 - Bastei::Parent, 55
 - Bastei::Root, 64
 - Init::Child, 21
- session_cap
 - Bastei::Session, 78
- session_control
 - Bastei::Session, 78
- SET_PAGER
 - Bastei::Cpu_session, 24
- set_pager
 - Bastei::Cpu_session, 25
- SIZE
 - Bastei::Dataspace, 26
- size
 - Bastei::Allocator_avl_base::Block, 11
 - Bastei::Buffer, 13
 - Bastei::Dataspace, 27
 - Fiasco::Msgbuf_base, 50
 - Pci::Device::Resource, 29
- size_dope
 - Fiasco::Msgbuf_base, 51
- Slab
 - Bastei::Slab, 81
- slab
 - Bastei::Slab_block, 83
- Slab_block
 - Bastei::Slab_block, 83
- slab_entry
 - Bastei::Slab_entry, 84
- slab_size
 - Bastei::Slab, 81
- sleep
 - Bastei::Client, 23
- sleep_forever
 - Bastei, 4
- Sliced_heap
 - Bastei::Sliced_heap, 85
- sp
 - Bastei::Thread_state, 91
- STACK
 - Nitpicker::View, 92
- stack
 - Nitpicker::View, 92
- START
 - Bastei::Cpu_session, 24
- start
 - Bastei::Cpu_session, 25

- Bastei::Server_activation, 66
- Bastei::Thread, 88
- STATE
 - Bastei::Cpu_session, 24
- state
 - Bastei::Cpu_session, 26
- sub_class
 - Pci::Device, 28
- Task
 - Bastei::Task, 86
- task_session
 - Bastei::Env, 31
- task_session/task_session.h, 108
- task_session_cap
 - Bastei::Task, 86
- Thread
 - Bastei::Thread, 88
- THREAD_NAME_LEN
 - Bastei::Cpu_session, 24
- Thread_state
 - Bastei::Thread_state, 90
- tid
 - Bastei::Thread_base, 90
 - Fiasco::Capability, 15
- Timer, 6
- Timer::Session, 78
 - ~Session, 79
 - MSLEEP, 79
 - msleep, 79
 - Opcode, 79
- timer_session/timer_session.h, 108
- TITLE
 - Nitpicker::View, 92
- title
 - Nitpicker::View, 92
- TRANSFER_QUOTA
 - Bastei::Parent, 55
 - Bastei::Ram_session, 57
- transfer_quota
 - Bastei::Child, 18
 - Bastei::Parent, 55
 - Bastei::Ram_session, 58
- Tslab
 - Bastei::Tslab, 91
- Type
 - Pci::Device::Resource, 29
- type
 - Pci::Device::Resource, 29
- UNLIMITED
 - Bastei::Heap, 32
- unlock
 - Bastei::Server_object, 69
- up
 - Bastei::Semaphore, 65
- USED
 - Bastei::Allocator_avl_base::Block, 10
 - Bastei::Ram_session, 57
- used
 - Bastei::Allocator_avl_base::Block, 11
 - Bastei::Ram_session, 59
- valid
 - Bastei::Session, 78
 - Fiasco::Capability, 15
- VIEWPORT
 - Nitpicker::View, 92
- viewport
 - Nitpicker::View, 92
- WAIT_FOR_IRQ
 - Bastei::Irq_session, 45
- wait_for_irq
 - Bastei::Irq_session, 46
- wait_for_service
 - Bastei::Service_pool, 71
- wakeup
 - Bastei::Client, 23
- WRITE
 - Bastei::Log_session, 48
- write
 - Bastei::Log_session, 48