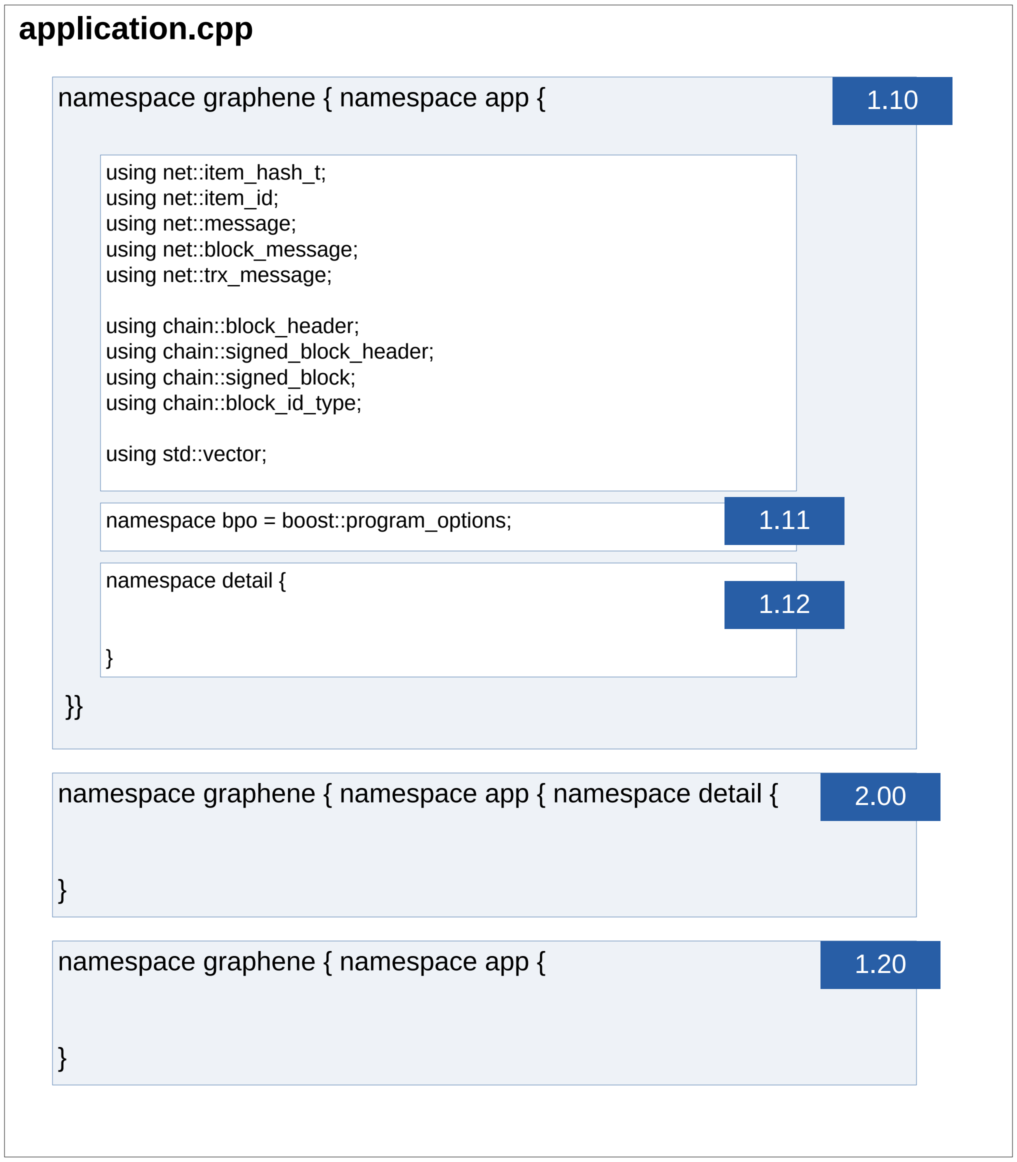
namespace structure



```
namespace graphene { namespace app {

    using net::item_hash_t;
    using net::item_id;
    using net::message;
    using net::block_message;
    using net::trx_message;

    using chain::block_header;
    using chain::signed_block_header;
    using chain::signed_block;
    using chain::block_id_type;

    using std::vector;

    namespace bpo = boost::program_options;

    namespace detail {

    }
}}

namespace graphene { namespace app { namespace detail {

}

namespace graphene { namespace app {

}
```

namespace graphene { namespace app { namespace detail {…}}

Items List

2.00	namespace graphene { namespace app { namespace detail {	
		void **application_impl::reset_p2p_node(const fc::path& data_dir)**
		std::vector<fc::ip::endpoint> **application_impl::resolve_string_to_ip_endpoints**(const std::string& endpoint_string)
		void **application_impl::new_connection**(const fc::http::websocket_connection_ptr& c)
		void **application_impl::reset_websocket_server**()
		void **application_impl::reset_websocket_tls_server**()
		void **application_impl::set_dbg_init_key**(graphene::chain::genesis_state_type& **genesis**, const std::string& **init_key**)
		void **application_impl::startup**()
		optional< api_access_info > application_impl::**get_api_access_info**(const string& username)const
		void application_impl::**set_api_access_info**(const string& username, api_access_info&& permissions)
		bool application_impl::**has_item**(const net::item_id& id)
		bool application_impl::**handle_block**(const graphene::net::block_message& blk_msg, bool sync_mode, std::vector<fc::uint160_t>& contained_transaction_message_ids)
		void application_impl:**:handle_transaction**(const graphene::net::trx_message& transaction_message)
		void application_impl::**handle_message**(const message& message_to_process)
		bool application_impl::**is_included_block**(const block_id_type& block_id)
		std::vector<item_hash_t> application_impl::**get_block_ids**(const std::vector<item_hash_t>& blockchain_synopsis, uint32_t& remaining_item_count, uint32_t limit)
		message application_impl::get_item(const item_id& id)
		chain_id_type application_impl:**:get_chain_id**() const
		std::vector<item_hash_t> application_impl::**get_blockchain_synopsis**(const item_hash_t& reference_point, uint32_t number_of_blocks_after_reference_point)
		void application_impl::**sync_status**(uint32_t item_type, uint32_t item_count)
		void application_impl::**connection_count_changed**(uint32_t c)
		uint32_t application_impl:**:get_block_number**(const item_hash_t& block_id)
		fc::time_point_sec application_impl::**get_block_time**(const item_hash_t& block_id)
		item_hash_t application_impl:**:get_head_block_id**() const
		uint32_t application_impl::**estimate_last_known_fork_from_git_revision_timestamp**(uint32_t unix_timestamp) const
		void application_impl::**error_encountered**(const std::string& message, const fc::oexception& error)
		uint8_t application_impl::**get_current_block_interval_in_seconds**() const
	} } } // namespace graphene namespace app namespace detail	