\libraries\app\api.cpp

```
#include <cctype>

#include <graphene/app/api.hpp>
#include <graphene/app/api_access.hpp>
#include <graphene/app/application.hpp>
#include <graphene/app/impacted.hpp>
#include <graphene/chain/database.hpp>
#include <graphene/chain/get_config.hpp>
#include <graphene/utilities/key_conversion.hpp>
#include <graphene/chain/protocol/fee_schedule.hpp>
#include <graphene/chain/confidential_object.hpp>
#include <graphene/chain/market_object.hpp>
#include <graphene/chain/transaction_object.hpp>
#include <graphene/chain/withdraw_permission_object.hpp>
#include <graphene/chain/worker_object.hpp>

#include <fc/crypto/hex.hpp>
#include <fc/smart_ref_impl.hpp>
#include <fc/thread/future.hpp>

namespace graphene { namespace app {
    login_api::login_api(application& a)
      :_app(a)
    login_api::~login_api()
    bool login_api::login(const string& user, const string& password)
    void login_api::enable_api( const std::string& api_name )
    block_api::block_api(graphene::chain::database& db) : _db(db) { }
    block_api::~block_api() { }
    vector<optional<signed_block>> block_api::get_blocks(uint32_t block_num_from, uint32_t block_num_to)const
    network_broadcast_api::network_broadcast_api(application& a):_app(a)
    void network_broadcast_api::on_applied_block( const signed_block& b )
    void network_broadcast_api::broadcast_transaction(const signed_transaction& trx)
    fc::variant network_broadcast_api::broadcast_transaction_synchronous(const signed_transaction& trx)
    void network_broadcast_api::broadcast_block( const signed_block& b )
    void network_broadcast_api::broadcast_transaction_with_callback(confirmation_callback cb, const signed_transaction& trx)
    network_node_api::network_node_api( application& a ) : _app( a )
    fc::variant_object network_node_api::get_info() const
    void network_node_api::add_node(const fc::ip::endpoint& ep)
    std::vector<net::peer_status> network_node_api::get_connected_peers() const
    std::vector<net::potential_peer_record> network_node_api::get_potential_peers() const
    fc::variant_object network_node_api::get_advanced_node_parameters() const
    void network_node_api::set_advanced_node_parameters(const fc::variant_object& params)
    fc::api<network_broadcast_api> login_api::network_broadcast()const
    fc::api<block_api> login_api::block()const
    fc::api<network_node_api> login_api::network_node()const
    fc::api<database_api> login_api::database()const
    fc::api<history_api> login_api::history() const
    fc::api<crypto_api> login_api::crypto() const
    fc::api<asset_api> login_api::asset() const
    fc::api<orders_api> login_api::orders() const
    fc::api<graphene::debug_witness::debug_api> login_api::debug() const
    vector<order_history_object> history_api::get_fill_order_history( asset_id_type a, asset_id_type b, uint32_t limit  )const
```

	vector<operation_history_object> history_api::**get_account_history**(account_id_type account, operation_history_id_type stop, unsigned limit, operation_history_id_type start) const
	vector<operation_history_object> history_api::**get_account_history_operations**(account_id_type account, int operation_id, operation_history_id_type start, operation_history_id_type stop, unsigned limit) const
	vector<operation_history_object> history_api::**get_relative_account_history**(account_id_type account, uint32_t stop, unsigned limit, uint32_t start) cons
	flat_set<uint32_t> history_api::**get_market_history_buckets**()const
	history_operation_detail history_api::**get_account_history_by_operations**(account_id_type account, vector<uint16_t> operation_types, uint32_t start, unsigned limit)
	vector<bucket_object> history_api::**get_market_history**(asset_id_type a, asset_id_type b, uint32_t bucket_seconds, fc::time_point_sec start, fc::time_point_sec end)const
	crypto_api::**crypto_api**(){};
	blind_signature crypto_api::**blind_sign**(const extended_private_key_type& key, const blinded_hash& hash, int i)
	signature_type crypto_api::**unblind_signature**(const extended_private_key_type& key, const extended_public_key_type& bob, const blind_signature& sig, const fc::sha256& hash, int i)
	commitment_type crypto_api::**blind**(const blind_factor_type& blind, uint64_t value)
	blind_factor_type crypto_api::**blind_sum**(const std::vector<blind_factor_type>& blinds_in, uint32_t non_neg)
	bool crypto_api::**verify_sum**(const std::vector<commitment_type>& commits_in, const std::vector<commitment_type>& neg_commits_in, int64_t excess)
	verify_range_result crypto_api::**verify_range**(const commitment_type& commit, const std::vector<char>& proof)
	std::vector<char> crypto_api::**range_proof_sign**(uint64_t min_value, const commitment_type& commit, const blind_factor_type& commit_blind, const blind_factor_type& nonce, int8_t base10_exp, uint8_t min_bits, uint64_t actual_value)
	verify_range_proof_rewind_result crypto_api::**verify_range_proof_rewind**(const blind_factor_type& nonce, const commitment_type& commit, const std::vector<char>& proof)
	range_proof_info crypto_api::**range_get_info**(const std::vector<char>& proof)
	asset_api::**asset_api**(graphene::chain::database& db) : _db(db) { }
	asset_api::**~asset_api()** { }
	vector<account_asset_balance> asset_api::**get_asset_holders**(asset_id_type asset_id, uint32_t start, uint32_t limit) const
	int asset_api::**:get_asset_holders_count**(asset_id_type asset_id) const
	vector<asset_holders> asset_api::**get_all_asset_holders**() const
	flat_set<uint16_t> orders_api::**get_tracked_groups**()const
	vector< limit_order_group > orders_api::**get_grouped_limit_orders**(asset_id_type base_asset_id, asset_id_type quote_asset_id, uint16_t group, optional<price> start, uint32_t limit)const
} } // graphene::app	
	7/9/2018
	BitShares Core Release 2.0.180612