# #

gRPC Gateway JSON REST

In irishub v1.0.0, the node continues to serve a REST server. However, the existing routes present in version v0.16.3 and earlier are now marked as deprecated, and new routes have been added via gRPC-gateway.

# #

API Port, Activation and Configuration

All routes are configured under the following fields in~/.iris/config/app.toml :

- api.enable = true|false
- field defines if the REST server should be enabled. Defaults totrue
- .
- api.address = {string}
- field defines the address (really, the port, since the host should be kept at0.0.0.0
- ) the server should bind to. Defaults totcp://0.0.0.0:1317
- .
- some additional API configuration options are defined in~/.iris/config/app.toml
- , along with comments, please refer to that file directly.

## #

gRPC-gateway REST Routes

If, for various reasons, you cannot use gRPC (for example, you are building a web application, and browsers don't support HTTP2 on which gRPC is built), then the IRIShub offers REST routes via gRPC-gateway.

gRPC-gatewayopen in new windowis a tool to expose gRPC endpoints as REST endpoints. For each RPC endpoint defined in a Protobuf service, the SDK offers a REST equivalent. For instance, querying token list could be done via the/irismod.token.Query/Tokens gRPC endpoint, or alternatively via the gRPC-gateway/irismod/token/tokens REST endpoint: both will return the same result. For each RPC method defined in a Protobuf service, the corresponding REST endpoint is defined as an option:

+++ https://github.com/irisnet/irismod/blob/master/proto/token/query.proto#L22

For application developers, gRPC-gateway REST routes needs to be wired up to the REST server, this is done by calling theRegisterGRPCGatewayRoutes function on the ModuleManager.

## #

Swagger

ASwaggeropen in new window(or OpenAPIv2) specification file is exposed under the/swagger route on the API server. Swagger is an open specification describing the API endpoints a server serves, including description, input arguments, return types and much more about each endpoint.

Enabling the/swagger endpoint is configurable inside~/.iris/config/app.toml via theapi.swagger field, which is set to true by default.

For application developers, you may want to generate your own Swagger definitions based on your custom modules. The IRIShub'sSwagger generation scriptopen in new windowis a good place to start.

# #

API Endpoints

IRIShub API Endpoints

API Endpoints Description Legacy REST Endpoint GET /cosmos/auth/v1beta1/accounts/{address} Return account details based on address GET /auth/accounts/{address} GET /cosmos/auth/v1beta1/params Query all parameters GET /cosmos/bank/v1beta1/balances/{address} Query the balance of all coins for a single account GET /bank/balances/{address} GET /cosmos/bank/v1beta1/balances/{address}/{denom} Query the balance of a single coin for a single account GET /cosmos/bank/v1beta1/denoms_metadata Query the client metadata for all registered coin denominations GET /cosmos/bank/v1beta1/denoms_metadata/{denom} Query the client metadata of a given coin denomination GET /cosmos/bank/v1beta1/params Query the parameters of bank module GET /cosmos/bank/v1beta1/supply Query the total supply of all coins GET /bank/total GET /cosmos/bank/v1beta1/supply/{denom} Query the supply of a single coin GET /bank/total/{denom} GET /cosmos/distribution/v1beta1/community_pool Query the community pool coins GET /distribution/community_pool GET /cosmos/distribution/v1beta1/delegators/{delegator_address}/rewards Query the total rewards accrued by each validator GET /distribution/delegators/{delegatorAddr}/rewards GET /cosmos/distribution/v1beta1/delegators/{delegator_address}/rewards/{validator_address} Query the total rewards accrued by a delegation GET /distribution/delegators/{delegatorAddr}/rewards/{validatorAddr} GET /cosmos/distribution/v1beta1/delegators/{delegator_address}/validators Query the validators of a delegator GET /cosmos/distribution/v1beta1/delegators/{delegator_address}/withdraw_address Query withdraw address of a delegator GET /distribution/delegators/{delegatorAddr}/withdraw_address GET /cosmos/distribution/v1beta1/params Query params of the distribution module GET /distribution/parameters GET /cosmos/distribution/v1beta1/validators/{validator_address}/commission Query accumulated

commission for a validator GET /cosmos/distribution/v1beta1/validators/{validator_address}/outstanding_rewards Query rewards of a validator address GET /distribution/validators/{validatorAddr}/outstanding_rewards GET /cosmos/distribution/v1beta1/validators/{validator_address}/slashes Query slash events of a validator GET /cosmos/evidence/v1beta1/evidence Query all evidence GET /cosmos/evidence/v1beta1/evidence/{evidence_hash} Query evidence based on evidence hash GET /cosmos/gov/v1beta1/params/{params_type} Query all parameters of the gov module GET /gov/parameters/{params_type} GET /cosmos/gov/v1beta1/proposals Query all proposals based on given status GET /gov/proposals GET /cosmos/gov/v1beta1/proposals/{proposal_id} Query proposal details based on ProposalID GET /gov/proposals/{proposal-id} GET /cosmos/gov/v1beta1/proposals/{proposal_id}/deposits Query all deposits of a single proposal GET /gov/proposals/{proposal-id}/deposits GET /cosmos/gov/v1beta1/proposals/{proposal_id}/deposits/{depositor} Query single deposit information based proposalID, depositAddr GET /gov/proposals/{proposal-id}/deposits/{depositor} GET /cosmos/gov/v1beta1/proposals/{proposal_id}/tally Query the tally of a proposal vote GET /gov/proposals/{proposal-id}/tally GET /cosmos/gov/v1beta1/proposals/{proposal_id}/votes Query votes of a given proposal GET /gov/proposals/{proposal-id}/votes GET /cosmos/gov/v1beta1/proposals/{proposal_id}/votes/{voter} Query voted information based on proposalID, voterAddr GET /gov/proposals/{proposal-id}/votes/{voter} GET /cosmos/params/v1beta1/params Query a specific parameter of a module, given its subspace and key GET /cosmos/slashing/v1beta1/params Query the parameters of slashing module GET /slashing/parameters GET /cosmos/slashing/v1beta1/signing_infos Query signing info of all validators GET /slashing/signing_infos GET /cosmos/slashing/v1beta1/signing_infos/{cons_address} Query the signing info of given cons address GET /cosmos/staking/v1beta1/delegations/{delegator_addr} Query all delegations of a given delegator address GET /staking/delegators/{delegatorAddr}/delegations GET /cosmos/staking/v1beta1/delegators/{delegator_addr}/redelegations Query redelegations of given address GET /staking/redelegations GET /cosmos/staking/v1beta1/delegators/{delegator_addr}/unbonding_delegations Query all unbonding delegations of a given delegator address GET /staking/delegators/{delegatorAddr}/unbonding_delegations GET /cosmos/staking/v1beta1/delegators/{delegator_addr}/validators Query all validators info for given delegator address GET /staking/delegators/{delegatorAddr}/validators GET /cosmos/staking/v1beta1/delegators/{delegator_addr}/validators/{validator_addr} Query validator info for given delegator validator pair GET /staking/delegators/{delegatorAddr}/validators/{validatorAddr} GET /cosmos/staking/v1beta1/historical_info/{height} Query the historical info for given height GET /cosmos/staking/v1beta1/params Query the staking parameters GET /staking/parameters GET /cosmos/staking/v1beta1/pool Query the pool info GET /staking/pool GET /cosmos/staking/v1beta1/validators Query all validators that match the given status GET /staking/validators GET /cosmos/staking/v1beta1/validators/{validator_addr} Query validator info for given validator address GET /staking/validators/{validatorAddr} GET /cosmos/staking/v1beta1/validators/{validator_addr}/delegations Query delegate info for given validator GET /staking/validators/{validatorAddr}/delegations GET /cosmos/staking/v1beta1/validators/{validator_addr}/delegations/{delegator_addr} Query delegate info for given validator delegator pair GET /staking/delegators/{delegatorAddr}/delegations/{validatorAddr} GET /cosmos/staking/v1beta1/validators/{validator_addr}/delegations/{delegator_addr}/unbonding_delegation Query unbonding info for given validator delegator pair GET /staking/delegators/{delegatorAddr}/unbonding_delegations/{validatorAddr} GET /cosmos/staking/v1beta1/validators/{validator_addr}/unbonding_delegations Query unbonding delegations of a validator GET /staking/validators/{validatorAddr}/unbonding_delegations GET /cosmos/upgrade/v1beta1/applied_plan/{name} Query a previously applied upgrade plan by its name GET /cosmos/upgrade/v1beta1/current_plan Query the current upgrade plan GET /cosmos/upgrade/v1beta1/upgraded_consensus_state/{last_height} Query the consensus state that will serve as a trusted kernel for the next version of this chain GET /ibc/core/channel/v1beta1/channels Query all the IBC channels of a chain GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id} Query an IBC channel GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/client_state Query for the client state for the channel associated with the provided channel identifiers GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/consensus_state/revision/{revision_number}/height/{revision_height} Query for the consensus state for the channel associated with the provided channel identifiers GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/next_sequence Return the next receive sequence for a given channel GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_acknowledgements Return all the packet acknowledgements associated with a channel GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_acks/{sequence} Query a stored packet acknowledgement hash GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_commitments Return all the packet commitments hashes associated with a channel GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_commitments/{packet_ack_sequences}/unreceived_acks Return all the unreceived IBC acknowledgements associated with a channel and sequences GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_commitments/{packet_commitment_sequences}/unreceived_packets Return all the unreceived IBC packets associated with a channel and sequences GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_commitments/{sequence} Query a stored packet commitment hash GET /ibc/core/channel/v1beta1/channels/{channel_id}/ports/{port_id}/packet_receipts/{sequence} Query if a given packet sequence has been received on the Queryd chain GET /ibc/core/channel/v1beta1/connections/{connection}/channels Query all the channels associated with a connection end GET /ibc/client/v1beta1/params Query all parameters of the ibc client GET /ibc/core/client/v1beta1/client_states Query all the IBC light clients of a chain GET /ibc/core/client/v1beta1/client_states/{client_id} Query an IBC light client GET /ibc/core/client/v1beta1/consensus_states/{client_id} Query all the consensus state associated with a given client GET /ibc/core/client/v1beta1/consensus_states/{client_id}/revision/{revision_number}/height/{revision_height} Query a consensus state associated with a client state at a given height GET /ibc/core/connection/v1beta1/client_connections/{client_id} Query the connection paths associated with a client state GET /ibc/core/connection/v1beta1/connections Query all the IBC connections of a chain GET /ibc/core/connection/v1beta1/connections/{connection_id} Query an IBC connection end GET /ibc/core/connection/v1beta1/connections/{connection_id}/client_state Query the client state associated with the connection GET /ibc/core/connection/v1beta1/connections/{connection_id}/consensus_state/revision/{revision_number}/height/{revision_height} Query the consensus state associated with the connection GET /ibc/applications/transfer/v1beta1/denom_traces Query all denomination traces GET /ibc/applications/transfer/v1beta1/denom_traces/{hash} Query a denomination trace information GET /ibc/applications/transfer/v1beta1/params Query all parameters of the ibc-transfer module GET /irismod/token/params Query the token parameters GET /irismod/token/tokens Return the token list GET /irismod/token/tokens/{denom} Return token with token name GET /irismod/token/tokens/{symbol}/fees Return the fees to issue or mint a token GET /irismod/token/total_burn Return all burnt coins GET /irismod/htlc/htlcs/{hash_lock} Query the HTLC by the specified hash lock GET /irismod/coinswap/liquidities/{denom} Return the total liquidity available for the provided denomination GET /irismod/nft/collections/{denom_id} Query the NFTs by the specified denom GET /irismod/nft/collections/{denom_id}/supply Query the total supply by a given denom GET /irismod/nft/denoms Query all the denoms GET /irismod/nft/denoms/{denom_id} Query the definition by a given denom ID GET /irismod/nft/nfts Query the NFTs by the specified owner GET /irismod/nft/nfts/{denom_id}/{token_id} Query the NFT by the given denom ID and token ID GET /irismod/service/bindings/{service_name} Return all service Bindings with service name and owner GET /irismod/service/bindings/{service_name}/{provider} Return service Binding with service name and provider GET /irismod/service/contexts/{request_context_id} Return the request context GET /irismod/service/definitions/{service_name} Return service

definition GET /irismod/service/fees/{provider} Return the earned service fee of one provider GET /irismod/service/owners/{owner}/withdraw-address Return the withdraw address of the binding owner GET /irismod/service/params Query the service parameters GET /irismod/service/requests/{request_context_id}/{batch_counter} Return all requests of one service call batch GET /irismod/service/requests/{request_id} Return the request GET /irismod/service/requests/{service_name}/{provider} Return all requests of one service with provider GET /irismod/service/responses/{request_context_id}/{batch_counter} Return all responses of one service call batch GET /irismod/service/responses/{request_id} Return the response of one request GET /irismod/service/schemas/{schema_name} Return the schema GET /irismod/oracle/feeds Query the feed list GET /irismod/oracle/feeds/{feed_name} Query the feed GET /irismod/oracle/feeds/{feed_name}/values Query the feed value GET /irismod/random/queue Query the random request queue GET /irismod/random/randoms/{req_id} Query the random result GET /irismod/record/records/{record_id} Query the record by the given record ID GET /irishub/mint/params Query the mint parameters GET /irishub/guardian/supers Return all Supers Tendermint API Endpoints

API Endpoints Description Legacy REST Endpoint GET /cosmos/base/tendermint/v1beta1/blocks/latest Return the latest block. GET /blocks/latest GET /cosmos/base/tendermint/v1beta1/blocks/{height} Query block for given height. GET /blocks/{height} GET /cosmos/base/tendermint/v1beta1/node_info Query the current node info. GET /node_info GET /cosmos/base/tendermint/v1beta1/syncing Query node syncing. GET /syncing GET /cosmos/base/tendermint/v1beta1/validatorsets/latest Query latest validator-set. GET /validatorsets/latest GET /cosmos/base/tendermint/v1beta1/validatorsets/{height} Query validator-set at a given height. GET /validatorsets/{height} POST /cosmos/tx/v1beta1/simulate Simulate executing a transaction for estimating gas usage. GET /cosmos/tx/v1beta1/txs Fetch txs by event. GET /txs POST /cosmos/tx/v1beta1/txs Broadcast transaction. POST /txs GET /cosmos/tx/v1beta1/txs/{hash} Fetch a tx by hash. GET /txs/{hash}

## #

Generating and Signing Transactions

It is not possible to generate or sign a transaction using REST, only to broadcast one. You can generating and signing transactions usinggRPC Client .

## #

Broadcasting Transactions

Broadcasting a transaction using the gRPC-gateway REST endpointcosmos/tx/v1beta1/txs can be done by sending a POST request as follows, where thetxBytes are the protobuf-encoded bytes of a signed transaction:

curl -X POST\ -H "Content-Type: application/json" \ -d'{"tx_bytes":"{{txBytes}}","mode":"BROADCAST_MODE_SYNC"}' \ "localhost:1317/cosmos/tx/v1beta1/txs"

## #

Querying Transactions

Querying transactions using the gRPC-gateway REST endpoint can be done by sending a GET request as follows:

- Query tx by hash:
- /cosmos/tx/v1beta1/txs/{hash}
- curl
- -X
- GET\
- -H
- "accept: application/json"
- \
- "http://localhost:1317/cosmos/tx/v1beta1/txs/{hash}"
- Query tx by events:
- /cosmos/tx/v1beta1/txs
- curl
- -X
- GET\
- -H
- "accept: application/json"
- \
- "http://localhost:1317/cosmos/tx/v1beta1/txs?events={event_content}"