Interface

Download this document from:
http://filenet.io/apkUrl/downloadZip?fileName=FN_API_EN.doc

Download fnv2wallets.exe from:
http://filenet.io/apkUrl/downloadZip?fileName=fnv2wallets.exe

Download fnv2wallets from:
http://filenet.io/apkUrl/downloadZip?fileName=fnv2wallets

Download latest block data:
http://filenet.io/apkUrl/downloadZip?fileName=fnv2.zip

First, run fnv2 wallet. (Windows: fnv2wallets.exe, linux: ./fnv2wallets)

API protocol of fnv2 uses http. Data format: json，post
http://HOSTNAME:50051/METHOD
return value{
    "error":string,
    "data":object
}

HOSTNAME is the host name where the PC wallet is located. METHOD can be the following methods.

1. Get the Latest Block Information

Query node server for block synchronization status	
Request POST	/synstat
Request parameter	{}
Response	data: { "newheight":uint64 *Latest block height* "curheight":uint64 *Height of the blocks that has been synchronized in this node* }

Sample code：

Request POST	http://localhost:50051/synstat
Request	{}

parameter	
curl	curl http://localhost:50051/synstat -X POST -d "{}"
Response：	{ "error": "", "data": { "newheight": 2157088, "curheight": 2153610 } }

2. Get Block

Query information of each block. Block information includes: transfer, approve, receipt, proxy	
Request POST	/getblock
Request parameter	{ "height": uint64 *Block height* }
Response	data: { "height":uint64 *Block height* "total":uint64 *The total amount of coins issued since block creation (unit 10^{-9} * FN . The same units below),* "reward":uint64 *Current block reward,* "timestamp":int64 Unix *Timestamp（unit: second）,* "transfercount":uint64 *transfer transaction amount,* "approvecount":uint64 *approve transaction amount,* "proxycount":uint64 *proxy transaction amount,* "rxcount":uin64 *receipt amount,* "seed":uint64 *Random number seed,* "txroot":string *Transaction root* "rxroot":string *Receipt root,* "xchash":string *Transaction partner hash,* "mienr":string *Current reward miner address,* "extra":string *Extension data hash,* "prevhash":string *Previous block hash,* "hash":string *Current block hash*, "sign":string *Current block signature,* "transfers":[{ "txid":string *Transaction ID,* "from":string *Transfer address,* "tocount":uint64 *Number of received addresses,* "value":uint64 *Amount（unit: 10^{-9} * FN）,* "timestamp":int64 *Timestamp，*

	"sign":string *Transaction signature，* "height":uint64 *Block height,* "transferdetails":[{ "to":string *Received address,* "value":uint64 *Received amount* }] }], "approves":[{ "txid":string *Transaction ID,* "from":string *Transfer address,* "tocount":uint64 *Number of received addresses,* "value":uint64 *Amount,* "timestamp":int64 *Timestamp，* "height":uint64 *Block height,* "sign":string *Transaction signature，* "expired":uint64 *Expiration time (block height calculation),* "expiredheight":uint64 *Expired block height,* "state":string（Three states: prepared, processing, complete）, "approvedetails":[{ "to":string *Available received address,* "minprop":uint64 Minimum ratio （ Range 0-10^{-9}, 10^{-9} represents 100%）, "maxprop":uint64 *Maximum ratio,* "accept":string *Receiver address,* "randprop":uint64 *Random ratio,* "value": *Amount obtained* }] }], "proxys::[{ "txid":string *Transaction ID,* "from":string *Transfer address,* "to":string *Proxy address,* "timestamp":int64 *Timestamp,* "expired":uint64 *Expire time,* "sign":string *Transaction signature,* "height":uint64 *Block height,* "expiredheight":uint64 *Expired height,* "state":string（prepared,processing,complete） }], "receipts":[{ "rxid":string *Receipt ID,* "txid": *Transaction ID,* "from":string *Receiver address,* "accept":bool *Whether to accept,*

	"timestamp"：int64 *Timestamp,* "sign":string *Signature,* "height":uint64 *Block height* }] }

Example code：

Request post	http://localhost:50051/getblock
Request parameter	{ "height":1620220 }
curl	curl http://localhost:50051/getblock -X POST -d '{"height":1620220}'
Response	{ "error": "", "data": { "height": 1620220, "total": 204463625000000000, "reward": 0, "timestamp": 1558927257, "transfercount": 0, "approvecount": 0, "proxycount": 0, "rxcount": 0, "seed": 897525801300417157, "txroot": "0000000000000000000000000000000000000000000000000000000000000000", "rxroot": "0000000000000000000000000000000000000000000000000000000000000000", "cxhash": "0000000000000000000000000000000000000000000000000000000000000000", "miner": "3Psbq3enwAmwXGa2QejWFdd9AwV1rczE6w1GPzs6WTPmJpKbmWghsLB ", "extra": "0000000000000000000000000000000000000000000000000000000000000000", "prevhash": "b68697f25e41aaa1af611246e33466ef3013386ceaea5011bc7b69aef881f272", "hash": "799ad39e879945a8238b06a1c76e450504bdf01f02b833a3e9ddbe459f35cbd2", "sign": "273166b708dc502292d4c7a87ad323db1022bee50f75395f682571a41c8cc0e50444 c6ee7739f80617ebc9524abc7c5ae3587c45493082f87cc4f2d63b756ae8" } }

3. Send Transfer Transaction

Send Transfer Transaction. Transfer can be one to many	
Request POST	/sendtransfer
Request parameter	{ "txid":string *Transaction ID,* "from":string *Transfer address,* "tocount":uint64 *Number of transferred addresses* "value":uint64 *Total amount transferred,* "timestamp":int64 *Timestamp (unit: second)* "sign":string *Signature,* "password":string *Password (Optional),* "transferdetails":[{ *//Transfer detail list. "to" can not be repeated, and is sorted by the order of "to"* "to":string *Transfer address,* "value":uint64 *Transferred amount of this address* }] }
	sign, txid values can be empty. Can use password for it. That is, you can choose either one: password or (txid, sign). When using password, the from wallet must be added to the local server by calling the addaccount interface.
Response	data: { "txid":string *Transaction id,* "height":uint64 *Block height* }

Example Code：

Request post	http://localhost:50051/sendtransfer
Request parameter	{ "from":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL", "tocount":1, "value":1, "timestamp":1574215319, "password":"xxxxxxxxx", "transferdetails":[{ "to":"3qpbcUefjq8ANdTnAPd1TF9SEiAGde16Pg2qXa8aQDVpKyD71K3LNPS", "value":1 }] }
curl：	curl http://localhost:50051/sendtransfer -X POST -d

	'{"from":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL","tocount":1,"value":1,"timestamp":1574215319,"password":"xxxxxxxxxxxxxxx","transferdetails":[{"to":"3qpbcUefjq8ANdTnAPd1TF9SEiAGde16Pg2qXa8aQDVpKyD71K3LNPS","value":1 }] }'
Response:	{ "error": "", "data": { "txid": "424ec19d3bbf6be5d16f6144b132c9df676767f84281818b7ba1ceb1c6b165c3", "height": 2996617 } }

4. Send Authorization Transaction

Send authorization transaction. This type of transaction refers to a special type of transfer transaction, that is, a from wallet initiates an authorization to one or more to wallets, and the to wallet divides the FNs in the authorization, and the amount of FNs obtained by each to wallet is random.	
Request POST	/sendapprove
Request parameter	{ "txid":string *Transaction ID,* "from":string *Transfer address,* "tocount":uint64 *Number of transferred addresses* "value":uint64 *Total amount transferred,* "timestamp":int64 *Timestamp (unit: second)* "expired":uint64 *The number of blocks that have expired* "sign":string *Signature,* "password":string *Password (Optional),* "approvedetails":[{ *//Transfer detail list. "to" can not be repeated, and is sorted by the order of "to"* "to":string *Available receiving address,* "minprop":uint64 *Minimum ratio,* "maxprop":uint64 *Maximum ratio* }] }

	How to calculate the amount of each recipient's due amount: the first recipient's remaining amount is the total amount of the transfer minus the amount received by the previous person, and the receiving probability is the first place in the list of minprop; maxprop randomly generates a ratio, and the received amount is the remaining total amount * the first random ratio. Then use the second random ratio, the third random ratio, and so on. sign, txid values can be empty. Can use password for it. That is, you can choose either one: password or (txid, sign). When using password, the from wallet must be added to the local server by calling the addaccount interface.
Response	data: { "txid":string *Transaction id,* "height":uint64 *Block height* }

Example Code:

Request post	http://localhost:50051/sendapprove
Request parameter	{ "from": "5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL", "tocount": 1, "value": 1, "timestamp": 1574217771, "expired": 2, "password": "xxxxxxxxxxxxxxx", "approvedetails": [{ "to": "3qpbcUefjq8ANdTnAPd1TF9SEiAGde16Pg2qXa8aQDVpKyD71K3LNPS", "minprop": 1, "maxprop": 2 }] }
curl	curl http://localhost:50051/sendapprove -X POST -d '{"from":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL","tocount":1,"value":1,"timestamp":1574217771,"expired":2,"password":"xxxxxxxxxxxxxx","approvedetails":[{"to":"3qpbcUefjq8ANdTnAPd1TF9SEiAGde16Pg2qXa8aQDVpKyD71K3LNPS","minprop":1,"maxprop":2}]}'
Response：	{ "error": "", "data": { "txid": "0e1b8fb99fb0683d6a17efa834ccfdfb1a622376d8b23cf589efc8e44d83c15b", "height": 2996852

	} }

5. Send Proxy Transaction

Send proxy transaction. This type of transaction is also called "voting", from the from wallet to the to wallet for voting. The to wallet is generally a super node wallet.	
Request POST	/sendproxy
Request parameter	{ "txid":string *Transaction id,* "from":string *Principal wallet address,* "to":string *Proxy wallet address,* "value":uint64 *Authorized amount,* "timestamp":int64 *Timestamp (unit: second)* "expired":uint64 *The number of blocks that have expired* "sign":string *Signature,* "password":string *Password (Optional)* }
	sign, txid values can be empty. Can use password for it. That is, you can choose either one: password or (txid, sign). When using password, the from wallet must be added to the local server by calling the addaccount interface.
Response	data: { "txid":string *Transaction id,* "height":uint64 *Block height* }

Example Code：

Request post	http://localhost:50051/sendproxy
Request parameter	{ "from": "5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL", "to": "3qpbcUefjq8ANdTnAPd1TF9SEiAGde16Pg2qXa8aQDVpKyD71K3LNPS", "value": 1, "timestamp": 1574218606, "expired": 2, "password": "xxxxxxxxxxxxx" }
curl	curl http://localhost:50051/sendproxy -X POST -d '{"from":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL","to":"3qpbcUefjq8ANdTnAPd1TF9SEiAGde16Pg2qXa8aQDVpKyD71K3

	LNPS","value":1,"timestamp":1574218606,"expired":2,"password":"xxxxxxxxxxxxx"}'
Response：	{ "error": "", "data": { "txid": "7b9b74f03858fef99ee69cf64c0a3e564160b8b589aa561ba845b83db0bc8ca2", "height": 2996932 } }

6. Send Receipt

Send Receipt. Authorized transaction approve. Feedback from the proxy of transaction proxy (Agree or Reject).	
Request POST	/sendreceipt
Request parameter	{ "rxid":string *Receipt id,* "txid":string *Transaction id, that is, txid of approve and proxy transaction* "from":string *Receiver address (that is "to" of approve and proxy transaction),* "accept":bool *Whether accept,* "timestamp":int64 *Timestamp (unit: second)* "sign":string *Signature,* "password":string *Password (Optional)* } *// rxid, sign values can be empty. Can use password for it.*
Response	data:{ "rxid":string *Receipt id,* "height":uint64 *Block height* }

Example Code：

Request Post	http://localhost:50051/sendreceipt
Request parameter	{ "txid": "424ec19d3bbf6be5d16f6144b132c9df676767f84281818b7ba1ceb1c6b165c3", "from": "5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL", "accept": true, "timestamp": 1574216934, "password": "xxxxxxxxxxxxxxxxx" }

curl	curl http://localhost:50051/sendreceipt -X POST -d '{"txid":"424ec19d3bbf6be5d16f6144b132c9df676767f84281818b7ba1ceb1c6b165c3","from":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL","accept":true,"timestamp":1574216934,"password":"xxxxxxxxxxxxxxxx"}'
Response：	{ "error": "", "data": { "rxid": "89744bbd61bd3fdeb6b065b5a3104b54e840e44eb71ffcfdbed31e84f2a1d306", "height": 2996762 } }

7. Get Balance

Get balance	
Request POST	/getbalance
Request parameter	{ "address":string *Wallet address* }
Response	data:{ "address":string *Wallet address,* "balance":uint64 *Transferable balance,* "approvebalance":uint64 *Authorized balance that has not been received,* "proxybalance":uint64 *Entrusted proxy balance,* "proxyaddress":uint64 *Proxy wallet address* }

Example Code：

Request post	http://localhost:50051/getbalance
Request Parameter	{ "address":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL " }
curl	curl http://localhost:50051/getbalance -X POST -d '{"address":"5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL "}'
Response：	{ "error": "", "data": { "address": "5vtot6JLACVJkPqZWSQqEZE3bT8DVRggQNngxGgU3pSvyG1DftosZNL ", "balance": 99224908766,

	"approvebalance": 0, "proxybalance": 0, "proxyaddress": "" } }

8. Add Account

Add fn wallet account to the node server	
Request POST	/addaccount
Request Parameter	{ "password":string *Password,* "privatekey":string *（64-bit hex string converted from a 32-byte private key）* }
	Generate a fn wallet on the node server with privatekey. The password of the wallet is: password. Privatekey can be any random string. A privatekey uniquely corresponds to an fn wallet address.
Response	data: { "address":string *Wallet address* }

Example Code：

Request post	http://localhost:50051/addaccount
Request Parameter	{ "password": "xxxxxxxxxxxxxxxxx", "privatekey": "0101010101101000110111011101010101001111011101111010101011010101" }
curl	curl http://localhost:50051/addaccount -X POST -d '{"password":"xxxxxxxxxxxxxxxxx","privatekey":"0010100101101000110111011010101010011110111011110101010110101010"}'
Response：	{ "error": "", "data": { "address": "5vtot6JLACVJkPqZWSQqaZE3Bt3DVRggQNngxTgU3pSvyG1EftosZNL" } }

9. Check Account List

View all wallet lists under this node server

Request	/listaccounts
Request POST	{}
Response	data:[{ "address":string *Wallet Address* }]

Example Code：

Request post	http://localhost:50051/listaccounts
Request Parameter	{}
curl	curl http://localhost:50051/listaccounts -X POST -d "{}"
Response：	{ "error": "", "data": { "addresslist": ["5vtot6JLACVJkPqZWSQqaZE3Bt3DVRggQNngxTgU3pSvyG1EftosZNL"] } }

10. Get Contribution Info

Get Contribution Info. Return the contribution information of all nodes in the whole network	
Request	/getcxinfo
Request Parameter POST	{ "height":uint64 *Block height,* }
	Some heights do not contribute information, so the result of the query *error* is not empty.
Response (ApiServer)	data: { "height":uint64 *Block height,* "json":{ "height":uint64 *Block height,* "cxhash":string *Contribution info hash,* "cbr":uint64 *Contribution value and margin equivalent conversion factor,* "agentcount":uint64 *Proxy amount,* "minercount":uint64 *Miner amount,* "agents":[{ *//Proxy List* "address":string *Proxy List,* "credit":float64 *Proxy credit,* "nodecount":uint64 *Select the total number of mining machines for this proxy node*

	"totalcontrib":uint64 *Select the total contribution of the mining machines of the proxy node* "validbalance":uint64 *Select the total effective margin of miners for this proxy node* "votenumber":uint64 *Select the total number of votes for miners of this agent node* }] "miners":[{ //*Miner List* "account":{ //*Miner account info* "address":string *Address,* "balance":uint64 *Available Balance,* "approvebalance":uint64 *Authorized but not received balance,* "proxybalance":uint64 *Proxy mining deposit,* "proxyaddress":uint64 *The miner's income (those select hosting mining service) is actually mined by the proxy address, but the mining machine binds its own address, so that the proxy can distribute the income according to the contribution of each miner* } "nodecount":uint64 *The number of mining machines bound by the miner,* "totalcontrib":uint64 *The total contribution of all mining machines of the miner,* "validbalance":uint64 *This miner's effective margin,* "credit":float64 *This miner's credit（If the miner uses hosting service, it is the proxy credit. If no hosting is used, then its value is 1）,* "votenumber":uint64 *The number of votes for the miner,* "contribs":[{ *//The mining machine's contribution list for each miner* "nodeId":string *Node id,* "address":string *Miner address,* "totalcap":uint64 *Total storage space,* "usedcap":uint64 *Actual storage space,* "integrity":float64 *Data integrity,* "onlinetime":uint64 *Online Time,* "contrib":uint64 *Contribution,* "timestamp":uint64 *Timestamp,* }] }] } }

Example Code：

Request post	http://localhost:50051/getcxinfo

Request Parameter	{ "height": 2921422 }
curl	curl http://localhost:50051/getcxinfo -X POST -d '{"height":2921422}'
Response：	{ "data": { "agentcount": 43, "agents": [{ "address": "3PsdC3kpfD1xAz46W9RYHFWBRsTfgu3FrGyQ6byPhJmQruoBCnQB31E", "credit": 3.8, "nodecount": 246, "totalcontrib": 838623161, "validbalance": 83837656100000, "votenumber": 318583093180000 }, ……], "cbr": 100000, "cxhash": "0000000000000000000000000000000000000000000000000000000000000000", "height": 2921422, "minercount": 883, "miners": [{ "account": { "address": "3PsdC3kpfD1xAz46W9RYHFWBRsTfgu3FrGyQ6byPhJmQruoBCnQB31E", "approvebalance": 0, "balance": 504910002000, "proxyaddress": "3PsdC3kpfD1xAz46W9RYHFWBRsTfgu3FrGyQ6byPhJmQruoBCnQB31E", "proxybalance": 97500000000000 }, "contribs": [{ "address": "", "contrib": 0, "integrity": 0, "nodeId": "QmNLncZ7rcgoir9NSN88ouMhoJeEfhVQxmNGKBfa5oVhis", "onlineTime": 0, "timestamp": 0, "totalCap": 0, "usedCap": 3775500 },

	……] }, ……] }, "error": "" }

11. Get Packaging Node Information

Get packing node information and return information list of all packing nodes.	
Request	/ getpacknode
Request parameter POST	{}
Response(ApiServer)	data: { "packNodes": [{ "address":string Packaging node address, "voteNumber": uint64 the number of votes obtained by packaging node, "height": uint64 统计时区块高度 the block height when counting votes }], "backUpNodes": [{ "address":string back up packaging node address 备份节点地址, "voteNumber": uint64 back up the number of votes obtained by packaging node "height": 2099389 the block height when counting votes }] }

Example Code：

Request post	http://localhost:50051/getpacknode
Request parameter	{}
curl	curl http://localhost:50051/getpacknode -X POST -d "{}"
Response：	{ "error": "", "data": { "packNodes": [{ "address": "0e584a18f94ebd32c04af41f8822c4e5697ceeaa", "voteNumber": 400001000100000,

	"height": 2140608 }], "backUpNodes": [{ "address": "b5576802a330051f74ec033474bafa2a5b246a26", "voteNumber": 10000000000, "height": 2140608 }, { "address": "84cf833149e521ce6f9d900dd9f3d3e843e8da4f", "voteNumber": 2000000000, "height": 2140608 }, { "address": "2df5599241f175c56418fa10e63310db8bbcbec2", "voteNumber": 2000000000, "height": 2140608 }] } }

12. Get transaction data

Get transaction data and get corresponding transaction information through txid. The transaction information needs to be synchronized to the block height of the transaction corresponding to txid .	
Request	/ gettransaction
Request parameter POST	{ "txid": string transaction id }
Response(ApiServer)	{ "error": "", "data": { "txid": string transaction id, "from": string transfer-out address, "tocount": uint64 the amount transferred to the address, "value": uint64 the total value transferred to the address， "timestamp": int64 timestamp, "transferdetails": [{ "to": string transfer-in address， "value": uint64 the value transferred to the address }], "sign":string signature, "height": uint64 block height } }

Example code：

Request post	http://localhost:50051/gettransaction
Request parameter	{ "txid":"e1e001f53d6a7b78830c7c4765be0766c117abd60566f154da4616e7738f2bde" }
curl	curl http://localhost:50051/gettransaction -X POST -d '{"txid":"e1e001f53d6a7b78830c7c4765be0766c117abd60566f154da4616e7738f2bde"}'
Response：	{ "error": "", "data": { "txid": "e1e001f53d6a7b78830c7c4765be0766c117abd60566f154da4616e7738f2bde", "from": "e0809a3e539a6fc93d1f26f86fe9801b9bfafc23", "tocount": 1, "value": 29947999900000, "timestamp": 1575973463, "transferdetails": [{ "to": "45579a7414f445406bf4a77c9c4c597725853ad4", "value": 29947999900000 }], "sign": "8f142083553ac609fb895e6436dbeef1e6e610155e9f2c31e78eb1c450a087cb6867fe42c1df15273afbf3040ac7fbca32440fc5051377e9798f28a6f0d2ba2c", "height": 1971439 } }

13. Send Votes

Send votes to the main chain.	
Request	/ sendvote
Request parameter POST	{ "txid": string transaction id }
Response (ApiServer)	{ "error": "", "data": null }

Example Code：

Request	http://localhost:50051/sendvote

post	
Request parameter	{ "txid":"e1e001f53d6a7b78830c7c4765be0766c117abd60566f154da4616e7738f2bde" }
curl	curl http://localhost:50051/sendvote -X POST -d '{"txid":"e1e001f53d6a7b78830c7c4765be0766c117abd60566f154da4616e7738f2bde"}'
Response：	{ "error": "", "data": null }

14. Close Wallet Program

（After sending this command, the program will close after a while and the program will exit normally.）	
Request POST	/6c45cb72a36e63d522aa54ed8adbd7a29a989474f2f77e0458af8800564ef3cb
Request Parameter	{}
Response	{}

Example Code：

Request post	http://localhost:50051/6c45cb72a36e63d522aa54ed8adbd7a29a989474f2f77e0458af8800564ef3cb
Request Parameter	{}
curl	curl http://localhost:50051/6c45cb72a36e63d522aa54ed8adbd7a29a989474f2f77e0458af8800564ef3cb -X POST -d "{}"
Response：	ok

15. Version Number

Version Number	
Request POST	/version
Request Parameter	{}
Response	{}

Example Code：

Request post	http://localhost:50051/version
Request Parameter	{}
curl	curl http://localhost:50051/version -X POST -d "{}"
Response：	FNV2.0.4