# JSON-RPC Interface

The headless daemon `bitcoind` has the JSON-RPC API enabled by default, the GUI `bitcoin-qt` has it disabled by default. This can be changed with the `-server` option. In the GUI it is possible to execute RPC methods in the Debug Console Dialog.

## Versioning

The RPC interface might change from one major version of Bitcoin Core to the next. This makes the RPC interface implicitly versioned on the major version. The version tuple can be retrieved by e.g. the `getnetworkinfo` RPC in `version`.

Usually deprecated features can be re-enabled during the grace-period of one major version via the `-deprecatedrpc=` command line option. The release notes of a new major release come with detailed instructions on what RPC features were deprecated and how to re-enable them temporarily.

## Security

The RPC interface allows other programs to control Bitcoin Core, including the ability to spend funds from your wallets, affect consensus verification, read private data, and otherwise perform operations that can cause loss of money, data, or privacy. This section suggests how you should use and configure Bitcoin Core to reduce the risk that its RPC interface will be abused.

- **Securing the executable:** Anyone with physical or remote access to the computer, container, or virtual machine running Bitcoin Core can compromise either the whole program or just the RPC interface. This includes being able to record any passphrases you enter for unlocking your encrypted wallets or changing settings so that your Bitcoin Core program tells you that certain transactions have multiple confirmations even when they aren't part of the best block chain. For this reason, you should not use Bitcoin Core for security sensitive operations on systems you do not exclusively control, such as shared computers or virtual private servers.

- **Securing local network access:** By default, the RPC interface can only be accessed by a client running on the same computer and only after the client provides a valid authentication credential (username and passphrase). Any program on your computer with access to the file system and local network can obtain this level of access. Additionally, other programs on your computer can attempt to provide an RPC interface on the same port as used by Bitcoin Core in order to trick you into revealing your authentication credentials. For this reason, it is important to only use Bitcoin Core for security-sensitive operations on a computer whose other programs you trust.

- **Securing remote network access:** You may optionally allow other computers to remotely control Bitcoin Core by setting the `rpcallowip` and `rpcbind` configuration parameters. These settings are only meant for enabling connections over secure private networks or connections that have been otherwise secured (e.g. using a VPN or port forwarding with SSH or stunnel). **Do not enable RPC connections over the public Internet.** Although Bitcoin Core's RPC interface does use authentication, it does not use encryption, so your login credentials are sent as clear text that can be read by anyone on your network path. Additionally, the RPC interface has not been hardened to withstand arbitrary Internet traffic, so changing the above settings to expose it to the Internet (even using something like a Tor onion service) could expose you to unconsidered vulnerabilities. See `bitcoind -help` for more information about these settings and other settings described in this document.

Related, if you use Bitcoin Core inside a Docker container, you may need to expose the RPC port to the host system. The default way to do this in Docker also exposes the port to the public Internet. Instead, expose it only on the host system's localhost, for example: `-p 127.0.0.1:8332:8332`

- **Secure authentication:** By default, Bitcoin Core generates unique login credentials each time it restarts and puts them into a file readable only by the user that started Bitcoin Core, allowing any of that user's RPC clients with read access to the file to login automatically. The file is `.cookie` in the Bitcoin Core configuration directory, and using these credentials is the preferred RPC authentication method. If you need to generate static login credentials for your programs, you can use the script in the `share/rpcauth` directory in the Bitcoin Core source tree. As a final fallback, you can directly use manually-chosen `rpcuser` and `rpcpassword` configuration parameters---but you must ensure that you choose a strong and unique passphrase (and still don't use insecure networks, as mentioned above).
- **Secure string handling:** The RPC interface does not guarantee any escaping of data beyond what's necessary to encode it as JSON, although it does usually provide serialized data using a hex representation of the bytes. If you use RPC data in your programs or provide its data to other programs, you must ensure any problem strings are properly escaped. For example, the `createwallet` RPC accepts arguments such as `wallet_name` which is a string and could be used for a path traversal attack without application level checks. Multiple websites have been manipulated because they displayed decoded hex strings that included HTML `<script>` tags. For this reason, and others, it is recommended to display all serialized data in hex form only.

## RPC consistency guarantees

State that can be queried via RPCs is guaranteed to be at least up-to-date with the chain state immediately prior to the call's execution. However, the state returned by RPCs that reflect the mempool may not be up-to-date with the current mempool state.

### Transaction Pool

The mempool state returned via an RPC is consistent with itself and with the chain state at the time of the call. Thus, the mempool state only encompasses transactions that are considered mine-able by the node at the time of the RPC.

The mempool state returned via an RPC reflects all effects of mempool and chain state related RPCs that returned prior to this call.

### Wallet

The wallet state returned via an RPC is consistent with itself and with the chain state at the time of the call.

Wallet RPCs will return the latest chain state consistent with prior non-wallet RPCs. The effects of all blocks (and transactions in blocks) at the time of the call is reflected in the state of all wallet transactions. For example, if a block contains transactions that conflicted with mempool transactions, the wallet would reflect the removal of these mempool transactions in the state.

However, the wallet may not be up-to-date with the current state of the mempool or the state of the mempool by an RPC that returned before this RPC. For example, a wallet transaction that was BIP-125-replaced in the mempool prior to this RPC may not yet be reflected as such in this RPC response.

## Limitations

There is a known issue in the JSON-RPC interface that can cause a node to crash if too many http connections are being opened at the same time because the system runs out of available file descriptors. To prevent this from happening you might want to increase the number of maximum allowed file descriptors in your system and try to prevent opening too many connections to your JSON-RPC interface at the same time if this is under your control. It is hard to give general advice since this depends on your system but if you make several hundred requests at once you are definitely at risk of encountering this issue.