Bitcoin Core version 0.10.0 is now available from:

https://bitcoin.org/bin/0.10.0/

This is a new major version release, bringing both new features and bug fixes.

Please report bugs using the issue tracker at github:

https://github.com/bitcoin/bitcoin/issues

Upgrading and downgrading

How to Upgrade

If you are running an older version, shut it down. Wait until it has completely shut down (which might take a few minutes for older versions), then run the installer (on Windows) or just copy over /Applications/Bitcoin-Qt (on Mac) or bitcoind/bitcoin-qt (on Linux).

Downgrading warning

Because release 0.10.0 makes use of headers-first synchronization and parallel block download (see further), the block files and databases are not backwards-compatible with older versions of Bitcoin Core or other software:

- Blocks will be stored on disk out of order (in the order they are received, really), which makes it incompatible with some tools or other programs. Reindexing using earlier versions will also not work anymore as a result of this.
- The block index database will now hold headers for which no block is stored on disk, which earlier versions won't support.

If you want to be able to downgrade smoothly, make a backup of your entire data directory. Without this your node will need start syncing (or importing from bootstrap.dat) anew afterwards. It is possible that the data from a completely synchronised 0.10 node may be usable in older versions as-is, but this is not supported and may break as soon as the older version attempts to reindex.

This does not affect wallet forward or backward compatibility.

Notable changes

Faster synchronization

Bitcoin Core now uses 'headers-first synchronization'. This means that we first ask peers for block headers (a total of 27 megabytes, as of December 2014) and validate those. In a second stage, when the headers have been discovered, we download the blocks. However, as we already know about the whole chain in advance, the blocks can be downloaded in parallel from all available peers.

In practice, this means a much faster and more robust synchronization. On recent hardware with a decent network link, it can be as little as 3 hours for an initial full synchronization. You may notice a slower progress in the very first few minutes, when headers are still being fetched and verified, but it should gain speed afterwards.

A few RPCs were added/updated as a result of this:

- `getblockchaininfo` now returns the number of validated headers in addition to the number of validated blocks.
- `getpeerinfo` lists both the number of blocks and headers we know we have in common with each peer. While synchronizing, the heights of the blocks that we have requested from peers (but haven't received yet) are also listed as 'inflight'.
- A new RPC `getchaintips` lists all known branches of the block chain, including those we only have headers for.

Transaction fee changes

This release automatically estimates how high a transaction fee (or how high a priority) transactions require to be confirmed quickly. The default settings will create transactions that confirm quickly; see the new 'txconfirmtarget' setting to control the tradeoff between fees and confirmation times. Fees are added by default unless the 'sendfreetransactions' setting is enabled.

Prior releases used hard-coded fees (and priorities), and would sometimes create transactions that took a very long time to confirm.

Statistics used to estimate fees and priorities are saved in the data directory in the `fee_estimates.dat` file just before program shutdown, and are read in at startup.

New command line options for transaction fee changes:

- `-txconfirmtarget=n` : create transactions that have enough fees (or priority) so they are likely to begin confirmation within n blocks (default: 1). This setting is over-ridden by the -paytxfee option.
- `-sendfreetransactions` : Send transactions as zero-fee transactions if possible (default: 0)

New RPC commands for fee estimation:

- `estimatefee nblocks` : Returns approximate fee-per-1,000-bytes needed for a transaction to begin confirmation within nblocks. Returns -1 if not enough transactions have been observed to compute a good estimate.
- `estimatepriority nblocks` : Returns approximate priority needed for a zero-fee transaction to begin confirmation within nblocks. Returns -1 if not enough free transactions have been observed to compute a good estimate.

RPC access control changes

Subnet matching for the purpose of access control is now done by matching the binary network address, instead of with string wildcard matching. For the user this means that `-rpcallowip` takes a subnet specification, which can be

- a single IP address (e.g. `1.2.3.4` or `fe80::0012:3456:789a:bcde`)
- a network/CIDR (e.g. `1.2.3.0/24` or `fe80::0000/64`)
- a network/netmask (e.g. `1.2.3.4/255.255.255.0` or `fe80::0012:3456:789a:bcde/ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff`)

An arbitrary number of `-rpcallow` arguments can be given. An incoming connection will be accepted if its origin address matches one of them.

For example:

0.9.x and before	0.10.x

`-rpcallowip=192.168.1.1`	`-rpcallowip=192.168.1.1` (unchanged)
`-rpcallowip=192.168.1.*`	`-rpcallowip=192.168.1.0/24`
`-rpcallowip=192.168.*`	`-rpcallowip=192.168.0.0/16`
`-rpcallowip=*` (dangerous!)	`-rpcallowip=::/0` (still dangerous!)

Using wildcards will result in the rule being rejected with the following error in debug.log:

```
Error: Invalid -rpcallowip subnet specification: *. Valid are a single IP (e.g.
1.2.3.4), a network/netmask (e.g. 1.2.3.4/255.255.255.0) or a network/CIDR (e.g.
1.2.3.4/24).
```

REST interface

A new HTTP API is exposed when running with the `-rest` flag, which allows unauthenticated access to public node data.

It is served on the same port as RPC, but does not need a password, and uses plain HTTP instead of JSON-RPC.

Assuming a local RPC server running on port 8332, it is possible to request:

- Blocks: http://localhost:8332/rest/block/*HASH*.*EXT*
- Blocks without transactions: http://localhost:8332/rest/block/notxdetails/*HASH*.*EXT*
- Transactions (requires `-txindex`): http://localhost:8332/rest/tx/*HASH*.*EXT*

In every case, *EXT* can be `bin` (for raw binary data), `hex` (for hex-encoded binary) or `json`.

For more details, see the `doc/REST-interface.md` document in the repository.

RPC Server "Warm-Up" Mode

The RPC server is started earlier now, before most of the expensive intialisations like loading the block index. It is available now almost immediately after starting the process. However, until all initialisations are done, it always returns an immediate error with code -28 to all calls.

This new behaviour can be useful for clients to know that a server is already started and will be available soon (for instance, so that they do not have to start it themselves).

Improved signing security

For 0.10 the security of signing against unusual attacks has been improved by making the signatures constant time and deterministic.

This change is a result of switching signing to use libsecp256k1 instead of OpenSSL. Libsecp256k1 is a cryptographic library optimized for the curve Bitcoin uses which was created by Bitcoin Core developer Pieter Wuille.

There exist attacks[1] against most ECC implementations where an attacker on shared virtual machine hardware could extract a private key if they could cause a target to sign using the same key hundreds of times. While using shared hosts and reusing keys are inadvisable for other reasons, it's a better practice to avoid the exposure.

OpenSSL has code in their source repository for derandomization and reduction in timing leaks that we've eagerly wanted to use for a long time, but this functionality has still not made its way into a released version of OpenSSL. Libsecp256k1 achieves significantly stronger protection: As far as we're aware this is the only deployed

implementation of constant time signing for the curve Bitcoin uses and we have reason to believe that libsecp256k1 is better tested and more thoroughly reviewed than the implementation in OpenSSL.

[1] https://eprint.iacr.org/2014/161.pdf

Watch-only wallet support

The wallet can now track transactions to and from wallets for which you know all addresses (or scripts), even without the private keys.

This can be used to track payments without needing the private keys online on a possibly vulnerable system. In addition, it can help for (manual) construction of multisig transactions where you are only one of the signers.

One new RPC, `importaddress`, is added which functions similarly to `importprivkey`, but instead takes an address or script (in hexadecimal) as argument. After using it, outputs credited to this address or script are considered to be received, and transactions consuming these outputs will be considered to be sent.

The following RPCs have optional support for watch-only: `getbalance`, `listreceivedbyaddress`, `listreceivedbyaccount`, `listtransactions`, `listaccounts`, `listsinceblock`, `gettransaction`. See the RPC documentation for those methods for more information.

Compared to using `getrawtransaction`, this mechanism does not require `-txindex`, scales better, integrates better with the wallet, and is compatible with future block chain pruning functionality. It does mean that all relevant addresses need to added to the wallet before the payment, though.

Consensus library

Starting from 0.10.0, the Bitcoin Core distribution includes a consensus library.

The purpose of this library is to make the verification functionality that is critical to Bitcoin's consensus available to other applications, e.g. to language bindings such as python-bitcoinlib or alternative node implementations.

This library is called `libbitcoinconsensus.so` (or, `.dll` for Windows). Its interface is defined in the C header bitcoinconsensus.h.

In its initial version the API includes two functions:

- `bitcoinconsensus_verify_script` verifies a script. It returns whether the indicated input of the provided serialized transaction correctly spends the passed scriptPubKey under additional constraints indicated by flags
- `bitcoinconsensus_version` returns the API version, currently at an experimental `0`

The functionality is planned to be extended to e.g. UTXO management in upcoming releases, but the interface for existing methods should remain stable.

Standard script rules relaxed for P2SH addresses

The IsStandard() rules have been almost completely removed for P2SH redemption scripts, allowing applications to make use of any valid script type, such as "n-of-m OR y", hash-locked oracle addresses, etc. While the Bitcoin protocol has always supported these types of script, actually using them on mainnet has been previously inconvenient as standard Bitcoin Core nodes wouldn't relay them to miners, nor would most miners include them in blocks they mined.

bitcoin-tx

It has been observed that many of the RPC functions offered by bitcoind are "pure functions", and operate independently of the bitcoind wallet. This included many of the RPC "raw transaction" API functions, such as createrawtransaction.

bitcoin-tx is a newly introduced command line utility designed to enable easy manipulation of bitcoin transactions. A summary of its operation may be obtained via "bitcoin-tx --help" Transactions may be created or signed in a manner similar to the RPC raw tx API. Transactions may be updated, deleting inputs or outputs, or appending new inputs and outputs. Custom scripts may be easily composed using a simple text notation, borrowed from the bitcoin test suite.

This tool may be used for experimenting with new transaction types, signing multi-party transactions, and many other uses. Long term, the goal is to deprecate and remove "pure function" RPC API calls, as those do not require a server round-trip to execute.

Other utilities "bitcoin-key" and "bitcoin-script" have been proposed, making key and script operations easily accessible via command line.

Mining and relay policy enhancements

Bitcoin Core's block templates are now for version 3 blocks only, and any mining software relying on its `getblocktemplate` must be updated in parallel to use libblkmaker either version 0.4.2 or any version from 0.5.1 onward. If you are solo mining, this will affect you the moment you upgrade Bitcoin Core, which must be done prior to BIP66 achieving its 951/1001 status. If you are mining with the stratum mining protocol: this does not affect you. If you are mining with the getblocktemplate protocol to a pool: this will affect you at the pool operator's discretion, which must be no later than BIP66 achieving its 951/1001 status.

The `prioritisetransaction` RPC method has been added to enable miners to manipulate the priority of transactions on an individual basis.

Bitcoin Core now supports BIP 22 long polling, so mining software can be notified immediately of new templates rather than having to poll periodically.

Support for BIP 23 block proposals is now available in Bitcoin Core's `getblocktemplate` method. This enables miners to check the basic validity of their next block before expending work on it, reducing risks of accidental hardforks or mining invalid blocks.

Two new options to control mining policy:

- `-datacarrier=0/1` : Relay and mine "data carrier" (OP_RETURN) transactions if this is 1.
- `-datacarriersize=n` : Maximum size, in bytes, we consider acceptable for "data carrier" outputs.

The relay policy has changed to more properly implement the desired behavior of not relaying free (or very low fee) transactions unless they have a priority above the AllowFreeThreshold(), in which case they are relayed subject to the rate limiter.

BIP 66: strict DER encoding for signatures

Bitcoin Core 0.10 implements BIP 66, which introduces block version 3, and a new consensus rule, which prohibits non-DER signatures. Such transactions have been non-standard since Bitcoin v0.8.0 (released in February 2013), but were technically still permitted inside blocks.

This change breaks the dependency on OpenSSL's signature parsing, and is required if implementations would want to remove all of OpenSSL from the consensus code.

The same miner-voting mechanism as in BIP 34 is used: when 751 out of a sequence of 1001 blocks have version number 3 or higher, the new consensus rule becomes active for those blocks. When 951 out of a sequence of 1001 blocks have version number 3 or higher, it becomes mandatory for all blocks.

Backward compatibility with current mining software is NOT provided, thus miners should read the first paragraph of "Mining and relay policy enhancements" above.

0.10.0 Change log

Detailed release notes follow. This overview includes changes that affect external behavior, not code moves, refactors or string updates.

RPC:

- `f923c07` Support IPv6 lookup in bitcoin-cli even when IPv6 only bound on localhost
- `b641c9c` Fix addnode "onetry": Connect with OpenNetworkConnection
- `171ca77` estimatefee / estimatepriority RPC methods
- `b750cf1` Remove cli functionality from bitcoind
- `f6984e8` Add "chain" to getmininginfo, improve help in getblockchaininfo
- `99ddc6c` Add nLocalServices info to RPC getinfo
- `cf0c47b` Remove getwork() RPC call
- `2a72d45` prioritisetransaction
- `e44fea5` Add an option `-datacarrier` to allow users to disable relaying/mining data carrier transactions
- `2ec5a3d` Prevent easy RPC memory exhaustion attack
- `d4640d7` Added argument to getbalance to include watchonly addresses and fixed errors in balance calculation
- `83f3543` Added argument to listaccounts to include watchonly addresses
- `952877e` Showing 'involvesWatchonly' property for transactions returned by 'listtransactions' and 'listsinceblock'. It is only appended when the transaction involves a watchonly address
- `d7d5d23` Added argument to listtransactions and listsinceblock to include watchonly addresses
- `f87ba3d` added includeWatchonly argument to 'gettransaction' because it affects balance calculation
- `0fa2f88` added includedWatchonly argument to listreceivedbyaddress/...account
- `6c37f7f` `getrawchangeaddress` : fail when keypool exhausted and wallet locked
- `ff6a7af` getblocktemplate: longpolling support
- `c4a321f` Add peerid to getpeerinfo to allow correlation with the logs
- `1b4568c` Add vout to ListTransactions output
- `b33bd7a` Implement "getchaintips" RPC command to monitor blockchain forks
- `733177e` Remove size limit in RPC client, keep it in server
- `6b5b7cb` Categorize rpc help overview
- `6f2c26a` Closely track mempool byte total. Add "getmempoolinfo" RPC
- `aa82795` Add detailed network info to getnetworkinfo RPC
- `01094bd` Don't reveal whether password is <20 or >20 characters in RPC
- `57153d4` rpc: Compute number of confirmations of a block from block height
- `ff36cbe` getnetworkinfo: export local node's client sub-version string
- `d14d7de` SanitizeString: allow '(' and ')'
- `31d6390` Fixed setaccount accepting foreign address
- `b5ec5fe` update getnetworkinfo help with subversion

- `ad6e601` RPC additions after headers-first
- `33dfbf5` rpc: Fix leveldb iterator leak, and flush before `gettxoutsetinfo`
- `2aa6329` Enable customising node policy for datacarrier data size with a -datacarriersize option
- `f877aaa` submitblock: Use a temporary CValidationState to determine accurately the outcome of ProcessBlock
- `e69a587` submitblock: Support for returning specific rejection reasons
- `af82884` Add "warmup mode" for RPC server
- `e2655e0` Add unauthenticated HTTP REST interface to public blockchain data
- `683dc40` Disable SSLv3 (in favor of TLS) for the RPC client and server
- `44b4c0d` signrawtransaction: validate private key
- `9765a50` Implement BIP 23 Block Proposal
- `f9de17e` Add warning comment to getinfo

Command-line options:

- `ee21912` Use netmasks instead of wildcards for IP address matching
- `deb3572` Add `-rpcbind` option to allow binding RPC port on a specific interface
- `96b733e` Add `-version` option to get just the version
- `1569353` Add `-stopafterblockimport` option
- `77cbd46` Let -zapwallettxes recover transaction meta data
- `1c750db` remove -tor compatibility code (only allow -onion)
- `4aaa017` rework help messages for fee-related options
- `4278b1d` Clarify error message when invalid -rpcallowip
- `6b407e4` -datadir is now allowed in config files
- `bdd5b58` Add option `-sysperms` to disable 077 umask (create new files with system default umask)
- `cbe39a3` Add "bitcoin-tx" command line utility and supporting modules
- `dbca89b` Trigger -alertnotify if network is upgrading without you
- `ad96e7c` Make -reindex cope with out-of-order blocks
- `16d5194` Skip reindexed blocks individually
- `ec01243` --tracerpc option for regression tests
- `f654f00` Change -genproclimit default to 1
- `3c77714` Make -proxy set all network types, avoiding a connect leak
- `57be955` Remove -printblock, -printblocktree, and -printblockindex
- `ad3d208` remove -maxorphanblocks config parameter since it is no longer functional

Block and transaction handling:

- `7a0e84d` ProcessGetData(): abort if a block file is missing from disk
- `8c93bf4` LoadBlockIndexDB(): Require block db reindex if any `blk*.dat` files are missing
- `77339e5` Get rid of the static chainMostWork (optimization)
- `4e0eed8` Allow ActivateBestChain to release its lock on cs_main
- `18e7216` Push cs_mains down in ProcessBlock
- `fa126ef` Avoid undefined behavior using CFlatData in CScript serialization
- `7f3b4e9` Relax IsStandard rules for pay-to-script-hash transactions
- `c9a0918` Add a skiplist to the CBlockIndex structure
- `bc42503` Use unordered_map for CCoinsViewCache with salted hash (optimization)
- `d4d3fbd` Do not flush the cache after every block outside of IBD (optimization)
- `ad08d0b` Bugfix: make CCoinsViewMemPool support pruned entries in underlying cache
- `5734d4d` Only remove actualy failed blocks from setBlockIndexValid

- `d70bc52` Rework block processing benchmark code
- `714a3e6` Only keep setBlockIndexValid entries that are possible improvements
- `ea100c7` Reduce maximum coinscache size during verification (reduce memory usage)
- `4fad8e6` Reject transactions with excessive numbers of sigops
- `b0875eb` Allow BatchWrite to destroy its input, reducing copying (optimization)
- `92bb6f2` Bypass reloading blocks from disk (optimization)
- `2e28031` Perform CVerifyDB on pcoinsdbview instead of pcoinsTip (reduce memory usage)
- `ab15b2e` Avoid copying undo data (optimization)
- `341735e` Headers-first synchronization
- `afc32c5` Fix rebuild-chainstate feature and improve its performance
- `e11b2ce` Fix large reorgs
- `ed6d1a2` Keep information about all block files in memory
- `a48f2d6` Abstract context-dependent block checking from acceptance
- `7e615f5` Fixed mempool sync after sending a transaction
- `51ce901` Improve chainstate/blockindex disk writing policy
- `a206950` Introduce separate flushing modes
- `9ec75c5` Add a locking mechanism to IsInitialBlockDownload to ensure it never goes from false to true
- `868d041` Remove coinbase-dependant transactions during reorg
- `723d12c` Remove txn which are invalidated by coinbase maturity during reorg
- `0cb8763` Check against MANDATORY flags prior to accepting to mempool
- `8446262` Reject headers that build on an invalid parent
- `008138c` Bugfix: only track UTXO modification after lookup

P2P protocol and network code:

- `f80cffa` Do not trigger a DoS ban if SCRIPT_VERIFY_NULLDUMMY fails
- `c30329a` Add testnet DNS seed of Alex Kotenko
- `45a4baf` Add testnet DNS seed of Andreas Schildbach
- `f1920e8` Ping automatically every 2 minutes (unconditionally)
- `806fd19` Allocate receive buffers in on the fly
- `6ecf3ed` Display unknown commands received
- `aa81564` Track peers' available blocks
- `caf6150` Use async name resolving to improve net thread responsiveness
- `9f4da19` Use pong receive time rather than processing time
- `0127a9b` remove SOCKS4 support from core and GUI, use SOCKS5
- `40f5cb8` Send rejects and apply DoS scoring for errors in direct block validation
- `dc942e6` Introduce whitelisted peers
- `c994d2e` prevent SOCKET leak in BindListenPort()
- `a60120e` Add built-in seeds for .onion
- `60dc8e4` Allow -onlynet=onion to be used
- `3a56de7` addrman: Do not propagate obviously poor addresses onto the network
- `6050ab6` netbase: Make SOCKS5 negotiation interruptible
- `604ee2a` Remove tx from AlreadyAskedFor list once we receive it, not when we process it
- `efad808` Avoid reject message feedback loops
- `71697f9` Separate protocol versioning from clientversion
- `20a5f61` Don't relay alerts to peers before version negotiation
- `b4ee0bd` Introduce preferred download peers

- `845c86d` Do not use third party services for IP detection
- `12a49ca` Limit the number of new addressses to accumulate
- `35e408f` Regard connection failures as attempt for addrman
- `a3a7317` Introduce 10 minute block download timeout
- `3022e7d` Require sufficent priority for relay of free transactions
- `58fda4d` Update seed IPs, based on bitcoin.sipa.be crawler data
- `18021d0` Remove bitnodes.io from dnsseeds.

Validation:

- `6fd7ef2` Also switch the (unused) verification code to low-s instead of even-s
- `584a358` Do merkle root and txid duplicates check simultaneously
- `217a5c9` When transaction outputs exceed inputs, show the offending amounts so as to aid debugging
- `f74fc9b` Print input index when signature validation fails, to aid debugging
- `6fd59ee` script.h: set_vch() should shift a >32 bit value
- `d752ba8` Add SCRIPT_VERIFY_SIGPUSHONLY (BIP62 rule 2) (test only)
- `698c6ab` Add SCRIPT_VERIFY_MINIMALDATA (BIP62 rules 3 and 4) (test only)
- `ab9edbd` script: create sane error return codes for script validation and remove logging
- `219a147` script: check ScriptError values in script tests
- `0391423` Discourage NOPs reserved for soft-fork upgrades
- `98b135f` Make STRICTENC invalid pubkeys fail the script rather than the opcode
- `307f7d4` Report script evaluation failures in log and reject messages
- `ace39db` consensus: guard against openssl's new strict DER checks
- `12b7c44` Improve robustness of DER recoding code
- `76ce5c8` fail immediately on an empty signature

Build system:

- `f25e3ad` Fix build in OS X 10.9
- `65e8ba4` build: Switch to non-recursive make
- `460b32d` build: fix broken boost chrono check on some platforms
- `9ce0774` build: Fix windows configure when using --with-qt-libdir
- `ea96475` build: Add mention of --disable-wallet to bdb48 error messages
- `1dec09b` depends: add shared dependency builder
- `c101c76` build: Add --with-utils (bitcoin-cli and bitcoin-tx, default=yes). Help string consistency tweaks. Target sanity check fix
- `e432a5f` build: add option for reducing exports (v2)
- `6134b43` Fixing condition 'sabotaging' MSVC build
- `af0bd5e` osx: fix signing to make Gatekeeper happy (again)
- `a7d1f03` build: fix dynamic boost check when --with-boost= is used
- `d5fd094` build: fix qt test build when libprotobuf is in a non-standard path
- `2cf5f16` Add libbitcoinconsensus library
- `914868a` build: add a deterministic dmg signer
- `2d375fe` depends: bump openssl to 1.0.1k
- `b7a4ecc` Build: Only check for boost when building code that requires it

Wallet:

- `b33d1f5` Use fee/priority estimates in wallet CreateTransaction
- `4b7b1bb` Sanity checks for estimates

- `c898846` Add support for watch-only addresses
- `d5087d1` Use script matching rather than destination matching for watch-only
- `d88af56` Fee fixes
- `a35b55b` Dont run full check every time we decrypt wallet
- `3a7c348` Fix make_change to not create half-satoshis
- `f606bb9` fix a possible memory leak in CWalletDB::Recover
- `870da77` fix possible memory leaks in CWallet::EncryptWallet
- `ccca27a` Watch-only fixes
- `9b1627d` [Wallet] Reduce minTxFee for transaction creation to 1000 satoshis
- `a53fd41` Deterministic signing
- `15ad0b5` Apply AreSane() checks to the fees from the network
- `11855c1` Enforce minRelayTxFee on wallet created tx and add a maxtxfee option

GUI:

- `c21c74b` osx: Fix missing dock menu with qt5
- `b90711c` Fix Transaction details shows wrong To:
- `516053c` Make links in 'About Bitcoin Core' clickable
- `bdc83e8` Ensure payment request network matches client network
- `65f78a1` Add GUI view of peer information
- `06a91d9` VerifyDB progress reporting
- `fe6bff2` Add BerkeleyDB version info to RPCConsole
- `b917555` PeerTableModel: Fix potential deadlock. #4296
- `dff0e3b` Improve rpc console history behavior
- `95a9383` Remove CENT-fee-rule from coin control completely
- `56b07d2` Allow setting listen via GUI
- `d95ba75` Log messages with type>QtDebugMsg as non-debug
- `8969828` New status bar Unit Display Control and related changes
- `674c070` seed OpenSSL PNRG with Windows event data
- `509f926` Payment request parsing on startup now only changes network if a valid network name is specified
- `acd432b` Prevent balloon-spam after rescan
- `7007402` Implement SI-style (thin space) thoudands separator
- `91cce17` Use fixed-point arithmetic in amount spinbox
- `bdba2dd` Remove an obscure option no-one cares about
- `bd0aa10` Replace the temporary file hack currently used to change Bitcoin-Qt's dock icon (OS X) with a buffer-based solution
- `94e1b9e` Re-work overviewpage UI
- `8bfdc9a` Better looking trayicon
- `b197bf3` disable tray interactions when client model set to 0
- `1c5f0af` Add column Watch-only to transactions list
- `21f139b` Fix tablet crash. closes #4854
- `e84843c` Broken addresses on command line no longer trigger testnet
- `a49f11d` Change splash screen to normal window
- `1f9be98` Disable App Nap on OSX 10.9+
- `27c3e91` Add proxy to options overridden if necessary
- `4bd1185` Allow "emergency" shutdown during startup
- `d52f072` Don't show wallet options in the preferences menu when running with -disablewallet

- `6093aa1` Qt: QProgressBar CPU-Issue workaround
- `0ed9675` [Wallet] Add global boolean whether to send free transactions (default=true)
- `ed3e5e4` [Wallet] Add global boolean whether to pay at least the custom fee (default=true)
- `e7876b2` [Wallet] Prevent user from paying a non-sense fee
- `c1c9d5b` Add Smartfee to GUI
- `e0a25c5` Make askpassphrase dialog behave more sanely
- `94b362d` On close of splashscreen interrupt verifyDB
- `b790d13` English translation update
- `8543b0d` Correct tooltip on address book page

Tests:

- `b41e594` Fix script test handling of empty scripts
- `d3a33fc` Test CHECKMULTISIG with m == 0 and n == 0
- `29c1749` Let tx (in)valid tests use any SCRIPT_VERIFY flag
- `6380180` Add rejection of non-null CHECKMULTISIG dummy values
- `21bf3d2` Add tests for BoostAsioToCNetAddr
- `b5ad5e7` Add Python test for -rpcbind and -rpcallowip
- `9ec0306` Add CODESEPARATOR/FindAndDelete() tests
- `75ebced` Added many rpc wallet tests
- `0193fb8` Allow multiple regression tests to run at once
- `92a6220` Hook up sanity checks
- `3820e01` Extend and move all crypto tests to crypto_tests.cpp
- `3f9a019` added list/get received by address/ account tests
- `a90689f` Remove timing-based signature cache unit test
- `236982c` Add skiplist unit tests
- `f4b00be` Add CChain::GetLocator() unit test
- `b45a6e8` Add test for getblocktemplate longpolling
- `cdf305e` Set -discover=0 in regtest framework
- `ed02282` additional test for OP_SIZE in script_valid.json
- `0072d98` script tests: BOOLAND, BOOLOR decode to integer
- `833ff16` script tests: values that overflow to 0 are true
- `4cac5db` script tests: value with trailing 0x00 is true
- `89101c6` script test: test case for 5-byte bools
- `d2d9dc0` script tests: add tests for CHECKMULTISIG limits
- `d789386` Add "it works" test for bitcoin-tx
- `df4d61e` Add bitcoin-tx tests
- `aa41ac2` Test IsPushOnly() with invalid push
- `6022b5d` Make `script_{valid,invalid}.json` validation flags configurable
- `8138cbe` Add automatic script test generation, and actual checksig tests
- `ed27e53` Add coins_tests with a large randomized CCoinViewCache test
- `9df9cf5` Make SCRIPT_VERIFY_STRICTENC compatible with BIP62
- `dcb9846` Extend getchaintips RPC test
- `554147a` Ensure MINIMALDATA invalid tests can only fail one way
- `dfeec18` Test every numeric-accepting opcode for correct handling of the numeric minimal encoding rule
- `2b62e17` Clearly separate PUSHDATA and numeric argument MINIMALDATA tests
- `16d78bd` Add valid invert of invalid every numeric opcode tests

- `f635269` tests: enable alertnotify test for Windows
- `7a41614` tests: allow rpc-tests to get filenames for bitcoind and bitcoin-cli from the environment
- `5122ea7` tests: fix forknotify.py on windows
- `fa7f8cd` tests: remove old pull-tester scripts
- `7667850` tests: replace the old (unused since Travis) tests with new rpc test scripts
- `f4e0aef` Do signature-s negation inside the tests
- `1837987` Optimize -regtest setgenerate block generation
- `2db4c8a` Fix node ranges in the test framework
- `a8b2ce5` regression test only setmocktime RPC call
- `daf03e7` RPC tests: create initial chain with specific timestamps
- `8656dbb` Port/fix txnmall.sh regression test
- `ca81587` Test the exact order of CHECKMULTISIG sig/pubkey evaluation
- `7357893` Prioritize and display -testsafemode status in UI
- `f321d6b` Add key generation/verification to ECC sanity check
- `132ea9b` miner_tests: Disable checkpoints so they don't fail the subsidy-change test
- `bc6cb41` QA RPC tests: Add tests block block proposals
- `f67a9ce` Use deterministically generated script tests
- `11d7a7d` [RPC] add rpc-test for http keep-alive (persistent connections)
- `34318d7` RPC-test based on invalidateblock for mempool coinbase spends
- `76ec867` Use actually valid transactions for script tests
- `c8589bf` Add actual signature tests
- `e2677d7` Fix smartfees test for change to relay policy
- `263b65e` tests: run sanity checks in tests too

Miscellaneous:

- `122549f` Fix incorrect checkpoint data for testnet3
- `5bd02cf` Log used config file to debug.log on startup
- `68ba85f` Updated Debian example bitcoin.conf with config from wiki + removed some cruft and updated comments
- `e5ee8f0` Remove -beta suffix
- `38405ac` Add comment regarding experimental-use service bits
- `be873f6` Issue warning if collecting RandSeed data failed
- `8ae973c` Allocate more space if necessary in RandSeedAddPerfMon
- `675bcd5` Correct comment for 15-of-15 p2sh script size
- `fda3fed` libsecp256k1 integration
- `2e36866` Show nodeid instead of addresses in log (for anonymity) unless otherwise requested
- `cd01a5e` Enable paranoid corruption checks in LevelDB >= 1.16
- `9365937` Add comment about never updating nTimeOffset past 199 samples
- `403c1bf` contrib: remove getwork-based pyminer (as getwork API call has been removed)
- `0c3e101` contrib: Added systemd .service file in order to help distributions integrate bitcoind
- `0a0878d` doc: Add new DNSseed policy
- `2887bff` Update coding style and add .clang-format
- `5cbda4f` Changed LevelDB cursors to use scoped pointers to ensure destruction when going out of scope
- `b4a72a7` contrib/linearize: split output files based on new-timestamp-year or max-file-size
- `e982b57` Use explicit fflush() instead of setvbuf()
- `234bfbf` contrib: Add init scripts and docs for Upstart and OpenRC

- `01c2807` Add warning about the merkle-tree algorithm duplicate txid flaw
- `d6712db` Also create pid file in non-daemon mode
- `772ab0e` contrib: use batched JSON-RPC in linarize-hashes (optimization)
- `7ab4358` Update bash-completion for v0.10
- `6e6a36c` contrib: show pull # in prompt for github-merge script
- `5b9f842` Upgrade leveldb to 1.18, make chainstate databases compatible between ARM and x86 (issue #2293)
- `4e7c219` Catch UTXO set read errors and shutdown
- `867c600` Catch LevelDB errors during flush
- `06ca065` Fix CScriptID(const CScript& in) in empty script case

Credits

Thanks to everyone who contributed to this release:

- 21E14
- Adam Weiss
- Aitor Pazos
- Alexander Jeng
- Alex Morcos
- Alon Muroch
- Andreas Schildbach
- Andrew Poelstra
- Andy Alness
- Ashley Holman
- Benedict Chan
- Ben Holden-Crowther
- Bryan Bishop
- BtcDrak
- Christian von Roques
- Clinton Christian
- Cory Fields
- Cozz Lovan
- daniel
- Daniel Kraft
- David Hill
- Derek701
- dexX7
- dllud
- Dominyk Tiller
- Doug
- elichai
- elkingtowa
- ENikS
- Eric Shaw
- Federico Bond
- Francis GASCHET
- Gavin Andresen
- Giuseppe Mazzotta
- Glenn Willen

- Gregory Maxwell
- gubatron
- HarryWu
- himynameismartin
- Huang Le
- Ian Carroll
- imharrywu
- Jameson Lopp
- Janusz Lenar
- JaSK
- Jeff Garzik
- JL2035
- Johnathan Corgan
- Jonas Schnelli
- jtimon
- Julian Haight
- Kamil Domanski
- kazcw
- kevin
- kiwigb
- Kosta Zertsekel
- LongShao007
- Luke Dashjr
- Mark Friedenbach
- Mathy Vanvoorden
- Matt Corallo
- Matthew Bogosian
- Micha
- Michael Ford
- Mike Hearn
- mrbandrews
- mruddy
- ntrgn
- Otto Allmendinger
- paveljanik
- Pavel Vasin
- Peter Todd
- phantomcircuit
- Philip Kaufmann
- Pieter Wuille
- pryds
- randy-waterhouse
- R E Broadley
- Rose Toomey
- Ross Nicoll
- Roy Badami
- Ruben Dario Ponticelli
- Rune K. Svendsen
- Ryan X. Charles
- Saivann

- sandakersmann
- SergioDemianLerner
- shshshsh
- sinetek
- Stuart Cardall
- Suhas Daftuar
- Tawanda Kembo
- Teran McKinney
- tm314159
- Tom Harding
- Trevin Hofmann
- Whit J
- Wladimir J. van der Laan
- Yoichi Hirai
- Zak Wilcox

As well as everyone that helped translating on Transifex.