Gnosis Guidelines Vault

# Changelog This changelog only contains changes starting from version 1.3.0 # Version 1.4.1 ## Compiler settings Solidity compiler: [0.7.6](<https://github.com/ethereum/solidity/releases/tag/v0.7.6>) (for more info see issue [#251](<https://github.com/safe-global/safe-contracts/issues/251>)) Solidity optimizer: `200 RUNS` ## Expected addresses with [0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552][singleton] ### Core contracts - `Safe` at `0x7951c7ef839e26F63DA87a42C9a87986507f1c07` - `SafeL2` at `0x245cc372C84B3645Bf0Ffe6538620B04a217988B` ### Factory contracts - `SafeProxyFactory` at `` ### Handler contracts - `TokenCallbackHandler` at `0xd5D82B6aDDc9027B22dCA772Aa68D5d74cdBdF44` - `CompatibilityFallbackHandler` at `0xf48f2B2d2a534e402487b3ee7C18c33Aec0Fe5e4` ### Lib contracts - `MultiSend` at `0xcEcC344Dd8426Ee2Ed226e64b8f4521751A2c1e1` - `MultiSendCallOnly` at `0xcEcC344Dd8426Ee2Ed226e64b8f4521751A2c1e1` - `CreateCall` at `0xcEcC344Dd8426Ee2Ed226e64b8f4521751A2c1e1` - `SignMessageLib` at `0xcb4524B63F2c532fF52FCbbBcd3C137D32c1D966` ### Storage reader contracts - `SimulateTxAccessor` at `0xa81636AaCE97783AAC45764B873793bE081dA592` ## Changes ### Bugfixes #### Remove `gasleft()` usage in `setupModules` Issue: [#568](<https://github.com/safe-global/safe-contracts/issues/568>) `setupModules` made a call to `gasleft()` that is invalid in the ERC-4337 standard. The call was replaced with `type(uint256).max` to forward all the available gas instead. # Version 1.4.0 ## Compiler settings Solidity compiler: [0.7.6](<https://github.com/ethereum/solidity/releases/tag/v0.7.6>) (for more info see issue [#251](<https://github.com/safe-global/safe-contracts/issues/251>)) Solidity optimizer: `YES` ## Expected addresses with [0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552][def] ### Core contracts - `Safe` at `0xc962E67D9490E154D81181879ddf4CD3b65D2132` - `SafeL2` at `0x1eb4681c549d995AbdC4aB189cAbb9f00B508cAb` ### Factory contracts - `SafeProxyFactory` at `0x4e1DCf7AD4e460CfD30791CCC4F9c8a4f820ec67` ### Handler contracts - `TokenCallbackHandler` at `0xeDCF620325E82e3B9836eaaeFdc4283E99Dd7562` - `CompatibilityFallbackHandler` at `0x2a15DE4410d4c8af0A7b6c12803120f43C42B820` ### Lib contracts - `MultiSend` at `0x38869bf66a61cF6bDB996A6aE40D5853Fd43B526` - `MultiSendCallOnly` at `0x9641d764fc13c8B624c04430C7356C1C7C8102e2` - `CreateCall` at `0x9b35Af71d77eaf8d7e40252370304687390A1A52` - `SignMessageLib` at `0x58FCe385Ed16beB4BCE49c8DF34c7d6975807520` ### Storage reader contracts - `SimulateTxAccessor` at `0x3d4BA2E0884aa488718476ca2FB8Efc291A46199` ## Changes ### General #### Drop "Gnosis" from contract names Removed the "Gnosis" prefix from all contract names. ### Core contract File: [`contracts/SafeL2.sol`](<https://github.com/safe-global/safe-contracts/blob/3c3fc80f7f9aef1d39aaae2b53db5f4490051b0d/contracts/SafeL2.sol>) #### Remove usage of the `GAS` opcode in module execute flows Issue: [#459](<https://github.com/safe-global/safe-contracts/issues/459>) The following rule of usage of the `GAS` opcode in the ERC-4337 standard made it impossible to build a module to support ERC4337: > - Must not use GAS opcode (unless followed immediately by one of { CALL, DELEGATECALL, CALLCODE, STATICCALL }.) We removed the `GAS` opcode usage in module transactions to forward all the available gas instead. #### Require the `to` address to be a contract in `setupModules` Issue: [#483](<https://github.com/safe-global/safe-contracts/issues/483>) The `setupModules` method was changed to require the `to` address to be a contract. If the `to` address is not a contract, the transaction will revert with a `GS002` error code. #### Enforce the `dataHash` is equal to `data` in the signature verification process for contract signatures Issue: [#497][def2] To prevent unexpected behaviour, the `dataHash` must now equal a hash of the `data` in the signature verification process for contract signatures. Otherwise, the transaction will revert with a `GS027` error code. #### Fix `getModulesPaginated` to return a correct `next` pointer Issue: [#461](<https://github.com/safe-global/safe-contracts/issues/461>) The `getModulesPaginated` method was fixed to return a correct `next` pointer. The `next` pointer now equals the last module in the returned array. #### Check the EIP-165 signature of the Guard before adding Issue: [#309](<https://github.com/safe-global/safe-contracts/issues/309>) When setting a guard, the core contract will check that the target address supports the Guard interface with an EIP-165 check. If it doesn't, the transaction will revert with the `GS300` error code. #### Index essential parameters when emitting events Issue: [#541](<https://github.com/safe-global/safe-contracts/issues/541>) Index essential parameters in the essential events, such as: - Owner additions and removals (Indexed parameter - owner address) - Fallback manager changes (Indexed parameter - fallback manager address) - Module additions and removals (Indexed parameter - module address) - Transaction guard changes (Indexed parameter - guard address) - Transaction execution/failure (Indexed parameter - transaction hash) ### Factory Umbrella issue: [#462](<https://github.com/safe-global/safe-contracts/issues/462>) #### Remove the `createProxy` method This method uses the `CREATE` opcode, which is not counterfactual for a specific deployment. This caused user errors and lost/stuck funds and is now removed. #### Add a check that Singleton exists for the initializer call If the initializer data is provided, the Factory now checks that the Singleton contract exists and the success of the call to avoid a proxy being deployed uninitialized #### Add `createChainSpecificProxyWithNonce` This method will use the chain id in the `CREATE2` salt; therefore, deploying a proxy to the same address on other networks is impossible. This method should enable the creation of proxies that should exist only on one network (e.g. specific governance or admin accounts) #### Remove the `calculateProxyAddress` method Method uses the revert approach to return data that only works well with some nodes, as they all return messages differently. Hence, we removed it, and the off-chain CREATE2 calculation is still possible. #### Remove the `proxyRuntimeCode` method The `.runtimeCode` method is not supported by the ZkSync compiler, so we removed it. ### Fallback handlers Files: - [0xf48f2B2d2a534e402487b3ee7C18c33Aec0Fe5e4](<https://github.com/safe-global/safe-contracts/blob/3c3fc80f7f9aef1d39aaae2b53db5f4490051b0d/contracts/handler/CompatibilityFallbackHandler.sol>) - [0xf48f2B2d2a534e402487b3ee7C18c33Aec0Fe5e4][def3] #### Rename `DefaultCallbackHandler` to `TokenCallbackHandler` Since the `DefaultCallbackHandler` only handled token callbacks, it was renamed to `TokenCallbackHandler`. #### Remove `NAME` and `VERSION` constants The `NAME` and `VERSION` constants were removed from the `CompatibilityFallbackHandler` contract. #### Fix function signature mismatch for `isValidSignature` Issue: [#440](<https://github.com/safe-global/safe-contracts/issues/440>) Fixed mismatch between the function signature in the `isValidSignature` method and the `ISignatureValidator` interface. ### Libraries #### CreateCall File: [`contracts/libraries/CreateCall.sol`](<https://github.com/safe-global/safe-contracts/blob/3c3fc80f7f9aef1d39aaae2b53db5f4490051b0d/contracts/libraries/CreateCall.sol>) #### Index the created contract address in the `ContractCreation` event Issue: [#541](<https://github.com/safe-global/safe-contracts/issues/541>) The deployed contract address in the `ContractCreation` event is now indexed. ### Deployment process #### Use the Safe Singleton Factory for all deployments Issue: [#460](<https://github.com/safe-global/safe-contracts/issues/460>) Deployments with the [Safe Singleton Factory][def] are now the default deployment process to ensure the same addresses on all chains. # Version 1.3.0-libs.0 ## Compiler settings Solidity compiler: [0.7.6](<https://github.com/ethereum/solidity/releases/tag/v0.7.6>) (more info see issue [#251](<https://github.com/safe-global/safe-contracts/issues/251>)) Solidity optimizer: `disabled` ## Expected addresses with [Deterministic Deployment Proxy](<https://github.com/Arachnid/deterministic-deployment-proxy>) (default) ### Core contracts - `GnosisSafe` at `0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552` - `GnosisSafeL2` at `0x3E5c63644E683549055b9Be8653de26E0B4CD36E` ### Factory contracts - `GnosisSafeProxyFactory` at `0xa6B71E26C5e0845f74c812102Ca7114b6a896AB2` ### Handler contracts - `DefaultCallbackHandler` at `0x1AC114C2099aFAf5261731655Dc6c306bFcd4Dbd` - `CompatibilityFallbackHandler` at `0xf48f2B2d2a534e402487b3ee7C18c33Aec0Fe5e4` ### Lib contracts - `MultiSend` at `0xA238CBeb142c10Ef7Ad8442C6D1f9E89e07e7761` - `MultiSendCallOnly` at `0x40A2aCCbd92BCA938b02010E17A5b8929b49130D` - `CreateCall` at `0x7cbB62EaA69F79e6873cD1ecB2392971036cFAa4` - `SignMessageLib` at `0xA65387F16B013cf2Af4605Ad8aA5ec25a2cbA3a2` ### Storage reader contracts - `SimulateTxAccessor` at `0x59AD6735bCd8152B84860Cb256dD9e96b85F69Da` ## Expected addresses with [Safe Singleton Factory][def] ### Core contracts - `GnosisSafe` at `0x69f4D1788e39c87893C980c06EdF4b7f686e2938` - `GnosisSafeL2` at `0xfb1bffC9d739B8D520DaF37dF666da4C687191EA` ### Factory contracts - `GnosisSafeProxyFactory` at `0xC22834581EbC8527d974F8a1c97E1bEA4EF910BC` ### Handler contracts - `DefaultCallbackHandler` at `0x3d8E605B02032A941Cfe26897Ca94d77a5BC24b3` - `CompatibilityFallbackHandler` at `0x017062a1dE2FE6b99BE3d9d37841FeD19F573804` ### Lib contracts - `MultiSend` at `0x998739BFdAAdde7C933B942a68053933098f9EDa` - `MultiSendCallOnly` at `0xA1dabEF33b3B82c7814B6D82A79e50F4AC44102B` - `CreateCall` at `0xB19D6FFc2182150F8Eb585b79D4ABcd7C5640A9d` - `SignMessageLib` at `0x98FFBBF51bb33A056B08ddf711f289936AafF717` ### Storage reader contracts - `SimulateTxAccessor` at `0x727a77a074D1E6c4530e814F89E618a3298FC044` ## Changes ### Deployment process To support deployment to networks that require replay protection support for the [Safe Singleton Factory][def] has been added. This will result in an additional set of deterministic addresses which are listed above. ### Libraries The following libraries have been marked as production ready. #### SignMessageLib File: [`contracts/libraries/SignMessage.sol`](<https://github.com/safe-global/safe-contracts/blob/e57df14ea96dc7dabf93f041c7531f2ab6755c76/contracts/libraries/SignMessage.sol>) Expected behaviour: The library is meant as a compatibility tool for the removed `signMessage` function from the pre-1.3.0 Safe contracts. It has the same signature and assumes the same storage layout as the previous Safe contract versions. After calling this function with a massage, the hash of that message should be marked as executed in the `signedMessages` mapping. #### GnosisSafeStorage File: [`contracts/libraries/GnosisSafeStorage.sol`](<https://github.com/safe-global/safe-contracts/blob/e57df14ea96dc7dabf93f041c7531f2ab6755c76/contracts/libraries/GnosisSafeStorage.sol>) Expected behaviour: The contract contains the basic storage layout of the `GnosisSafe.sol` contract and can be used by library contracts to access the storage variables. # Version 1.3.0 ## Compiler settings Solidity compiler: [0.7.6](<https://github.com/ethereum/solidity/releases/tag/v0.7.6>) (more info see issue [#251](<https://github.com/safe-global/safe-contracts/issues/251>)) Solidity optimizer: `disabled` ## Expected deterministic deployment addresses ### Core contracts - `GnosisSafe` at `0xd9Db270c1B5E3Bd161E8c8503c55cEABeE709552` - `GnosisSafeL2` at `0x3E5c63644E683549055b9Be8653de26E0B4CD36E` ### Factory contracts - `GnosisSafeProxyFactory` at `0xa6B71E26C5e0845f74c812102Ca7114b6a896AB2` ### Handler contracts - `DefaultCallbackHandler` at `0x1AC114C2099aFAf5261731655Dc6c306bFcd4Dbd` - `CompatibilityFallbackHandler` at `0xf48f2B2d2a534e402487b3ee7C18c33Aec0Fe5e4` ### Lib contracts - `MultiSend` at `0xA238CBeb142c10Ef7Ad8442C6D1f9E89e07e7761` - `MultiSendCallOnly` at `0x40A2aCCbd92BCA938b02010E17A5b8929b49130D` - `CreateCall` at `0x7cbB62EaA69F79e6873cD1ecB2392971036cFAa4` ### Storage reader contracts - `SimulateTxAccessor` at `0x59AD6735bCd8152B84860Cb256dD9e96b85F69Da` ## Changes ### Core contract File: [`contracts/GnosisSafe.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/GnosisSafe.sol>) #### Add chainId to transaction hash Issue: [#170](<https://github.com/safe-global/safe-contracts/issues/170>) Expected behaviour: The `chainId` has been added to the [EIP-712](<https://eips.ethereum.org/EIPS/eip-712>) domain. In case of a change of the `chainId` (e.g. hardfork related) the new `chainId` will automatically be used for future signature checks. #### Add transaction guard Issue: [#224](<https://github.com/safe-global/safe-contracts/issues/224>) Expected behaviour: It is possible to add a transaction guard, which can check all of the parameters that have been sent to `execTransaction` prior to execution. For this check the `checkTransaction` needs to be implemented by the guard. In case that `checkTransaction` reverts, `execTransaction` will also revert. Another check that can be implemented by the guard is `checkAfterExecution`. This check is called at the very end of the execution and allows to perform checks on the final state of the Safe. The parameters passed to that check are the `safeTxHash` and a `success` boolean. #### Add StorageAccessible support Issue: [#201](<https://github.com/safe-global/safe-contracts/issues/201>) Expected behaviour: It is possible to use `simulateDelegatecallInternal` to simulate logic on the Safe by providing a contract and calldata. This contract will then be called via a delegatecall and the result will be returned via a revert.The revert data will have the following format: `success:bool || response.length:uint256 || response:bytes`. Important: This method will always revert. #### Remove changeMasterCopy Expected behaviour: It is not possible anymore to change the singleton address (formerly known as master copy) via a method call. To make the implications of a singleton address change more visible it is required to use a delegatecall with a migration contract. (See example migration in libraries) #### Make checkSignature public Issue: [#248](<https://github.com/safe-global/safe-contracts/issues/248>) Expected behaviour: The `checkSignature` method is now a view method that is public. This makes it possible that it can be used in other contracts (e.g. modules) to make it easier to reuse existing signature check logic. The function expects that there are at least enough valid signatures to hit the threshold. Another method that has been added to make the usage from external contracts easier is `checkNSignatures` which allows to set how many valid signatures are expected. Note: The storage allocated by `approveHash` will no longer be zeroed when being used in `checkSignature`. If this is required a delegatecall with a contract that zeroes past approved hashes should be used. #### Remove authorized from requiredTxGas Issue: [#247](<https://github.com/safe-global/safe-contracts/issues/247>) Expected behaviour: To make it easier to interact with this method (e.g. by providing a wrapper). The requirement that the method can only be called by the Safe itself has been removed. The method will still always revert. Note: This method is superseded by the `StorageAccessible` logic and will be removed in the next major version. #### Move EIP-1271 logic to fallback handler Issue: [#223](<https://github.com/safe-global/safe-contracts/issues/223>) Expected behaviour: As [EIP-1271](<https://eips.ethereum.org/EIPS/eip-1271>) is still changing the logic for it has been moved to a fallback handler. The fallback handler uses the `checkSignatures` method to validate the signatures. Also this fallback handler supports the latest version of [EIP-1271]([https://eips.ethereum.org/EIPS/eip-1271](https://eips.ethereum.org/EIPS/eip-1271" \t "_blank)). The logic to mark a message hash as signed in the contract also has been moved to other contracts. `getMessageHash` has been moved to a fallback handler and `signMessage` into a library that can be used via delegatecall. Note: The `checkSignature` method still uses the previous version of [EIP-1271](<https://eips.ethereum.org/EIPS/eip-1271>) that uses the data to be signed instead of the hash of the data. #### Send along msg.sender to fallback handler Issue: [#246](<https://github.com/safe-global/safe-contracts/issues/246>) Expected behaviour: When the Safe forwards a call to the fallback handler it will append the `msg.sender` to the calldata. This will allow the fallback handler to use this information. Note: Fallback handlers should make sure that the connected Safe supports this, else this can be used by the caller to influence the fallback handler (by specifying an arbitrary `msg.sender`) #### Revert on failure if safeTxGas and gasPrice are 0 Issue: [#274](<https://github.com/safe-global/safe-contracts/issues/274>) Expected behaviour: If `safeTxGas` is 0 (therefore all available gas has been used for the internal tx) and `gasPrice` is also 0 (therefore no refund is involved) the transaction will revert when the internal tx fails. This makes it easier to interact with the Safe without having to estimate the internal transaction ahead of time. #### Add setup event Issue: [#233](<https://github.com/safe-global/safe-contracts/issues/233>) Expected behaviour: The Safe now emits an event that contains all setup information that influences the State of the nearly setup Safe. The initializer calldata is omitted to prevent excessive gas costs. And the refund information is omitted as they don’t have an influence on the internal contract state. #### Add incoming ETH event Issue: [#209](<https://github.com/safe-global/safe-contracts/issues/209>) Expected behaviour: When the Safe is receiving ETH it will now trigger an event (with exception of ETH received via a call to `execTransaction` or as a result of a selfdestruct of another contract). Note: It will not be possible anymore to send ETH via the solidity calls transfer or send to a Safe. This is expected to break because of the gas costs changes with the Berlin hard fork ([EIP-2929](<https://eips.ethereum.org/EIPS/eip-2929>)) in any case (even without the event) when using the legacy transaction format. As there is also a new transaction format ([EIP-2930](<https://eips.ethereum.org/EIPS/eip-2930>)) it is possible to use that together with the correct access list to still execute transfer/ send calls and emit the event. ### Layer 2 #### Add contract version that emits Safe tx information via events File: [`contracts/GnosisSafeL2.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/GnosisSafeL2.sol>) Expected behaviour: The extended version will emit an event with all the information related to the Safe transaction that will be executed. As this is quite gas expensive, it is only expected that this version will be used on Layer 2 networks with low gas prices. It is expected that the events are emitted on entry to the method. As the normal Safe methods emit already some events after the execution of the Safe transaction. This will make it possible to connect other events to that call as they are "boxed" by the GnosisSafeL2 events and the GnosisSafe events. Example: On entry into `execTransaction` of the `GnosisSafeL2` contract a `SafeMultiSigTransaction` event will be emitted that contains all the parameters of the function and the `nonce`, `msg.sender` and `threshold`. Once the internal execution has finished the `execTransaction` of the `GnosisSafe` contract will emit a `ExecutionSuccess` or `ExecutionFailure` event. When processing the events of that transaction it is now possible to connect all events that were emitted between these two events to this specific Safe transaction. Same can be done with the `SafeModuleTransaction` and `ExecutionFromModuleSuccess` (or `ExecutionFromModuleFailure`) events when executing a transaction via a module. ### Fallback handlers #### Add EIP-165 support to DefaultCallbackHandler Issue: [#161](<https://github.com/safe-global/safe-contracts/issues/161>) File: [`contracts/handler/DefaultCallbackHandler.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/handler/DefaultCallbackHandler.sol>) Expected behaviour: Indicate via the `supportsInterface` method of [EIP-165](<https://eips.ethereum.org/EIPS/eip-165>) that the [EIP-721](<https://eips.ethereum.org/EIPS/eip-721>) and [EIP-1155](<https://eips.ethereum.org/EIPS/eip-1155>) receiver interfaces are supported. #### Add CompatibilityFallbackHandler Issue: [#223](<https://github.com/safe-global/safe-contracts/issues/223>) File: [`contracts/handler/CompatibilityFallbackHandler.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/handler/CompatibilityFallbackHandler.sol>) Expected behaviour: The `CompatibilityFallbackHandler` extends the `DefaultCallbackHandler` and implements support for some logic that has been removed from the core contracts. Namely [EIP-1271]([https://eips.ethereum.org/EIPS/eip-1271](https://eips.ethereum.org/EIPS/eip-1271" \t "_blank)) support and the non reverting method of the `StorageAccessible` contract. Also the fallback manager contains the logic to verify Safe messages. #### Add possibility to get sender in fallback handler File: [`contracts/handler/HandlerContext.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/handler/HandlerContext.sol>) Expected behaviour: The `HandlerContext` can be used to retrieve the `msg.sender` and the Safe (aka manager) that have been forwarding the call to the fallback handler. The `msg.sender` is expected to be appended to the calldata (e.g. last 20 bytes). This will only work if used with a Safe contract that supports this (e.g. 1.3.0 or newer). ### Guard #### Add DelegateCallTransactionGuard File: [`contracts/examples/guards/DelegateCallTransactionGuard.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/examples/guards/DelegateCallTransactionGuard.sol>) Note: \*\*This contract is meant as an example to demonstrate how to facilitate a guard. This should not be used in production without further checks.\*\* Expected behaviour: This transaction guard can be used to prevent that Safe transactions that use a delegatecall operation are being executed. It is also possible to specify an exception when deploying the contract (e.g. a `MultiSendCallOnly` instance). #### Add DebugTransactionGuard File: [`contracts/examples/guards/DebugTransactionGuard.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/examples/guards/DebugTransactionGuard.sol>) Note: \*\*This contract is meant as an example to demonstrate how to facilitate a guard. This should not be used in production without further checks.\*\* Expected behaviour: This transaction guard can be used to log more details about a transaction. This is similar to what the L2 version of the Safe does, but implemented as a transaction guard. One event will be emitted containing the transaction details and another to track the status of a specific nonce. #### Add ReentrancyTransactionGuard File: [`contracts/examples/guards/ReentrancyTransactionGuard.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/examples/guards/ReentrancyTransactionGuard.sol>) Note: \*\*This contract is meant as an example to demonstrate how to facilitate a guard. This should not be used in production without further checks.\*\* Expected behaviour: This transaction guard can be used to prevent that Safe transactions can re-enter the `execTransaction` method. The transaction guard does not differentiate between different Safes, so if multiple Safes use the same guard instance it prevents entrancy in all of the connected Safes. ### Libraries #### Make multiSend payable to avoid check on msg.value Issue: [#227](<https://github.com/safe-global/safe-contracts/issues/227>) File: [`contracts/libraries/MultiSend.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/libraries/MultiSend.sol>) Expected behaviour: The `multiSend` is now payable therefore will enforce anymore that `msg.value` is 0. ETH that is not transferred out again will remain in `this` (the calling contract when used via a delegatecall or the contract when used via call, only possible with `MultiSendCallOnly`) #### Add MuliSend that disallows delegate operation File: [`contracts/libraries/MultiSendCallOnly.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/libraries/MultiSendCallOnly.sol>) Expected behaviour: The logic is the same as for the normal `MultiSend`, but when an attempt is made to execute a transaction via a delegatecall the contract will revert. Note: The encoding of the data send to the `multiSend` method is exactly the same as for the normal `MultiSend`, this makes it easy to exchange the contracts depending on the use case. #### Add base contract for Safe storage layout File: [`contracts/examples/libraries/GnosisSafeStorage.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/examples/libraries/GnosisSafeStorage.sol>) Note: \*\*This contract is meant as an example to demonstrate how to access the Safe state within a library contract. This should not be used in production without further checks.\*\* Expected behaviour: The contract contains the basic storage layout of the `GnosisSafe.sol` contract. #### Add contract to mark Safe messages as signed File: [`contracts/examples/libraries/SignMessage.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/examples/libraries/SignMessage.sol>) Note: \*\*This contract is meant as an example to demonstrate how to mark Safe message as signed in the signedMessages mapping. This should not be used in production without further checks.\*\* Expected behaviour: The library is meant as a compatibility tool for the removed `signMessage` function from the pre-1.3.0 Safe contracts. It has the same signature and assumes the same storage layout as the previous Safe contract versions. After calling this function with a massage, the hash of that message should be marked as executed in the `signedMessages` mapping. #### Add Migration example to downgrade from 1.3.0 to 1.2.0 File: [`contracts/examples/libraries/Migrate\_1\_3\_0\_to\_1\_2\_0.sol`](<https://github.com/safe-global/safe-contracts/blob/ad6c7355d5bdf4f7fa348fbfcb9f07431769a3c9/contracts/examples/libraries/Migrate_1_3_0_to_1_2_0.sol>) Note: \*\*This contract is meant as an example to demonstrate how to facilitate migration in the future. This should not be used in production without further checks.\*\* Expected behaviour: This migration can be used to migrate a Safe to another singleton address. Once the migration has been executed the singleton address will point to the address specified in the constructor of the migration and the domain separator will be properly set in storage (as this is required by the 1.2.0 version of the Safe contracts). Note: This is meant as an example contract, only to be used in production if you know what you do. [def]: <https://github.com/safe-global/safe-singleton-factory> [def2]: <https://github.com/safe-global/safe-contracts/issues/497> [def3]: <https://github.com/safe-global/safe-contracts/blob/3c3fc80f7f9aef1d39aaae2b53db5f4490051b0d/contracts/handler/TokenCallbackHandler.sol>