Argent Vault on Ethereum L1

How to make sure your dapp works seamlessly with Argent!

What is Argent Vault?

Argent is the first smart wallet for Ethereum. It's the only non-custodial mobile wallet that combines easy access to dapps and security features such as recovery without seed phrases, trusted contacts and multisig. These features are made possible by Argent's smart contract architecture.

Argent website:https://www.argent.xyz

Smart Contracts Repository:https://github.com/argentlabs/argent-contracts/

What is WalletConnect?

WalletConnect is an open source protocol for connecting decentralized applications ("dapps") to mobile wallets, via QR code scanning or deep linking. A user can interact securely with any dapp from their mobile phone, making WalletConnect-enabled wallets a safer choice compared to desktop or browser extension wallets.

WalletConnect and Argent

As a mobile wallet, Argent fully supports the WalletConnect protocol. Since Argent is a smart contract based wallet, you need to pay attention to some specific features.

Checking the correct signature

Externally Owned Accounts (EOA) can sign messages with their associated private keys, however, smart contracts cannotEIP-1271 outlines a standard way for contracts to verify if a provided signature is valid when an account is a contract.

The Argent wallet implements theisValidSignature() method, as per EIP-1271. A dapp that wants to verify the signature of an Argent user should therefore callisValidSignature() on the Argent wallet instead ofecrecover() (as would be used to verify signatures from EOA accounts).

```

Copy contractERC1271{

// bytes4(keccak256("isValidSignature(bytes32,bytes)") bytes4 constant internalMAGICVALUE=0x1626ba7e;

/ *@devShould return whether the signature provided is valid for the provided data*@param_hash EIP-191 compliant hash of the message*@param_signature Signature byte array associated with _data * * MUST return the bytes4 magic value 0x1626ba7e when function passes. * MUST NOT modify state (using STATICCALL for solc < 0.5, view modifier for solc > 0.5) * MUST allow external calls* / functionisValidSignature( bytes32 _hash, bytes memory _signature ) public view returns(bytes4 magicValue); }

```

The parameter_hash should beEIP-191 compliant. See full example here in JavaScript (usingethers.js library):

```

Copy constargentABI=[ 'function isValidSignature(bytes32 _message, bytes _signature) public view returns (bool)' ];

constwalletAddress="0x..."; constmessage="Lorem ipsum dolor sit amet";

constargentWallet=newethers.Contract(walletAddress,argentABI,provider); consthashMessage=ethers.utils.hashMessage(message);

try{ constreturnValue=awaitargentWallet.isValidSignature(hashMessage,signature) }catch(error) { // signature is not valid }

```

Wallet detection

We have developed and deployed a simple contract to detect if a given address corresponds to an Argent wallet. The contract exposes a singleisArgentWallet(address) method that returns true if the code deployed at the input address matches a deployed version of the Argent wallet.

```

Copy contract ArgentWalletDetector {

/ *@noticeChecks if an address is an Argent wallet*@param_wallet The target wallet */ functionisArgentWallet(address _wallet)externalviewreturns(bool);

}

```

The detector contract is deployed on Ropsten testnet and Mainnet:

```

Copy Ropsten: 0xF230cF8980BaDA094720C01308319eF192F0F311 Mainnet: 0xeca4B0bDBf7c55E9b7925919d03CbF8Dc82537E8

```

Multicall

Argent wallets support the ability to batch transactions into a single transaction, for example an ERC20 approval followed by a contract call.

```

Copy constargentABI=[ "function isValidSignature(bytes32 _message, bytes _signature) public view returns (bool)", "function wc_multiCall((address to, uint256 value, bytes data)[] _transactions)", ];

constwalletAddress="0x...";

constargentWallet=newethers.Contract(walletAddress,argentABI,provider);
```

constresult=awaitargentWallet.wc_multiCall([// approve USDC { to:"0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48", value:0, data: "0x095ea7b3000000000000000000000000def1c0ded9bec7f1a1670819833240f027b25eff0000000000000000000000000000000000000000ffffffffffff096fb4da20000", }, // call contract { to:"0x7a250d5630b4cf539739df2c5dacb4c659f2488d", value:0, data:"0x.....", },]);

```

The best practice is to check if the wallet is an Argent wallet using the wallet detector contract and then batch the transactions:

```

Copy constwalletDetectorAddress="0xeca4B0bDBf7c55E9b7925919d03CbF8Dc82537E8"; constwalletDetectorABI=["function isArgentWallet(address _wallet) external view returns (bool)"];

constwalletDetector=newethers.Contract(walletDetectorAddress,walletDetectorABI,provider); constisArgent=awaitwalletDetector.isArgentWallet(address);

if(isArgent) { // do a multicall }else{ // do single calls }

```

Meta transactions and relayers

Argent wallet uses the concept of meta transactions. These are a particular type of transaction which are signed by one or more key pairs (in our case the wallet's owner and potentially their guardians) but are submitted to the Ethereum network by a relayer. The relayer pays the gas fee (in ETH) and the wallet will refund the relayer (in ETH or ERC20 tokens) up to an amount signed by the wallet's owner.

From the dapp's perspective, this is managed by the Argent mobile application. The dapp will submit a regular{ to, value, data } transaction to the web3 provider. This transaction will be transmitted to the Argent mobile application through WalletConnect. The mobile wallet will transform the data into a meta transaction:

- to
- will be the ArgentRelayerManager
- contract address
- data
- will be the encoded data of the call to theexecute()
- method with the relevant parameters
-

The dapp will receive the transaction hash in order to monitor the status of the transaction and events will be emitted as usual.

Our relayer has the ability to replay a transaction with a higher gas price due to fluctuating network conditions. The transaction hash is modified and the dapp will not be aware of the new transaction hash.

One solution could be for the dapp to observe a specific event being emitted instead of transaction status. We are working on defining a standard for a dapp to be notified when a transaction is replaced and the transaction hash changes (contact us if you are interested in this).

Login UX best practice

To improve the login experience for Argent users, we recommend showing an Argent branded login button at the top level. This ensures people less familiar with WalletConnect can log in to your app. When clicking on the Argent login, you then present the WalletConnect QR code.

You can find an example of this implementation onhttps://app.flexa.network/connect

?

FAQ

Can I use the ERC20 permit function?

It depends on how the ERC20 smart contract implements thepermit() method. If you look below (sourceEtherscan ) at the Maker DAI implementation, you will notice at line 24 the use ofecrecover() and not EIP-1271 to verify the signature. Here, the call will fail with Argent wallets.

```

Copy functionpermit(address holder, address spender, uint256 nonce, uint256 expiry, bool allowed, uint8 v, bytes32 r, bytes32 s)external{ bytes32 digest= keccak256(abi.encodePacked("\x19\x01", DOMAIN_SEPARATOR, keccak256(abi.encode(PERMIT_TYPEHASH, holder, spender, nonce, expiry, allowed))));

require(holder!=address(0),"Dai/invalid-address-0"); require(holder==ecrecover(digest,v,r,s),"Dai/invalid-permit"); require(expiry==0||now<=expiry,"Dai/permit-expired"); require(nonce==nonces[holder]++,"Dai/invalid-nonce"); uint wad=allowed?uint(-1):0; allowance[holder][spender]=wad; emitApproval(holder,spender,wad); }

```

I'm asking for ERC20 approval of an amount of1 0 53 10^{53} 1 0 53 but it doesn't seem to work

Argent may override the "infinite value" and ask a user to input a smaller amount due to the additional security risk of allowing large persistent approvals. When checking the allowance in your dapp you need to check for the exact amount the user wants to spend.

Can I use 3rd party relayers with Argent?

Yes, but the relayer needs to implement a call to theexecute() method of ourRelayerManager smart contract. You will have to get the wallet owner signing the correct payload which is not straightforward. Please contact us if you're interested in this topic.

Do you estimate gas?

Yes, we useeth_estimateGas before sending any transaction.

Do you support WalletConnect mobile deeplinks?

Yes, WalletConnect will work on mobile browsers (or native mobile apps) and will deep link to the Argent wallet.

How can I test the integration?

You can test Argent on Ropsten on both iOS and Android. Contact dapps@argent.xyz to receive a test build.

Previous Perkz Next Get in touch
```

Last updated3 months ago