get-available-tokens)

Was this helpful? Export as PDF

Token Management

Request all available tokens and their balances, manage token approvals and more.

Get available tokens

getTokens

Retrieves a list of all available tokens on specified chains.

Parameters

- params
- (TokensRequest, optional): Configuration for the requested tokens.
-
    - chains
-
    - (ChainId[], optional): List of chain IDs or keys. If not specified, returns tokens on all available chains.
-
    - chainTypes
-
    - (ChainType[], optional): List of chain types.
- options
- (RequestOptions, optional): Additional request options.

Returns

A Promise that resolves toTokensResponse .

Example

```

Copy import{ ChainType,getTokens }from'@lifi/sdk';

try{ consttokens=awaitgetTokens({ chainTypes:[ChainType.EVM,ChainType.SVM], }); console.log(tokens); }catch(error) { console.error(error); }

```

getToken

Fetches details about a specific token on a specified chain.

Parameters

- chain
- (ChainKey | ChainId): ID or key of the chain that contains the token.
- token
- (string): Address or symbol of the token on the requested chain.
- options

- (RequestOptions, optional): Additional request options.

Returns

A Promise that resolves toToken object.

Example

```

Copy import{ getToken }from'@lifi/sdk';

constchainId=1; consttokenAddress='0x0000000000000000000000000000000000000000';

try{ consttoken=awaitgetToken(chainId,tokenAddress); console.log(token); }catch(error) { console.error(error); }

```

Get token balance

Please ensure that you configure the SDK with EVM/Solana providers first. They are required to use this functionality. Additionally, it is recommended to provide your private RPC URLs, as public ones are used by default and may rate limit you for multiple requests, such as getting the balance of multiple tokens at once.

Read moreConfigure SDK Providers .

getTokenBalance

Returns the balance of a specific token a wallet holds.

Parameters

- walletAddress
- (string): A wallet address.
- token
- (Token): A Token object.

Returns

A Promise that resolves to aTokenAmount ornull .

Example

```

Copy import{ getToken,getTokenBalance }from'@lifi/sdk';

constchainId=1; consttokenAddress='0x0000000000000000000000000000000000000000'; constwalletAddress='0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0';

try{ consttoken=awaitgetToken(chainId,tokenAddress); consttokenBalance=awaitgetTokenBalance(walletAddress,token); console.log(tokenBalance); }catch(error) { console.error(error); }

```

getTokenBalances

Returns the balances for a list of tokens a wallet holds.

Parameters

- walletAddress
- (string): A wallet address.
- tokens
- (Token[]): A list of Token objects.

Returns

A Promise that resolves to a list ofTokenAmount objects.

Example

```
```

Copy import{ ChainId,getTokenBalances,getTokens }from'@lifi/sdk';

constwalletAddress='0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0';

try{ consttokensResponse=awaitgetTokens(); constoptimismTokens=tokensResponse.tokens[ChainId.OPT]; consttokenBalances=awaitgetTokenBalances(walletAddress,optimismTokens); console.log(tokenBalances); }catch(error) { console.error(error); }

```

getTokenBalancesByChain

Queries the balances of tokens for a specific list of chains for a given wallet.

Parameters

- walletAddress
- (string): A wallet address.
- tokensByChain
- ({ [chainId: number]: Token[] }): A list of Token objects organized by chain IDs.

Returns

A Promise that resolves to an object containing the tokens and their amounts on different chains.

Example

```

Copy import{ getTokenBalancesByChain }from'@lifi/sdk';

constwalletAddress='0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0'; consttokensByChain={ 1:[{ chainId:1, address:'0x6B175474E89094C44Da98b954EedeAC495271d0F', symbol:'DAI', name:'DAI Stablecoin', decimals:18, priceUSD:'0.9999', },], 10:[{ chainId:10, address:'0x4200000000000000000000000000000000000042', symbol:'OP', name:'Optimism', decimals:18, priceUSD:'1.9644', },], };

try{ constbalances=awaitgetTokenBalancesByChain(walletAddress,tokensByChain); console.log(balances); }catch(error) { console.error(error); }

```

Managing token allowance

Token allowance and approval functionalities are specific to EVM (Ethereum Virtual Machine) chains. It allows smart contracts to interact with ERC-20 tokens by approving a certain amount of tokens that a contract can spend from the user's wallet.

Please ensure that you configure the SDK with the EVM provider. It is required to use this functionality.

Read more[Configure SDK Providers](Configure SDK Providers) .

getTokenAllowance

Fetches the current allowance for a specific token.

Parameters

- token
- (BaseToken): The token for which to check the allowance.
- ownerAddress
- (string): The owner of the token.
- spenderAddress
- (string): The spender address that was approved.

Returns

A Promise that resolves to abigint representing the allowance or undefined if the token is a native token.

Example

```

Copy import{ getTokenAllowance }from'@lifi/sdk';

consttoken={ address:'0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48', chainId:1, };

constownerAddress='0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0';
constspenderAddress='0x1231DEB6f5749EF6cE6943a275A1D3E7486F4EaE';

try{ constallowance=awaitgetTokenAllowance(token,ownerAddress,spenderAddress); console.log('Allowance:',allowance);
}catch(error) { console.error('Error:',error); }

```

getTokenAllowanceMulticall

Fetches the current allowance for a list of token/spender address pairs.

Parameters

- ownerAddress
- (string): The owner of the tokens.
- tokens
- (TokenSpender[]): A list of token and spender address pairs.

Returns

A Promise that resolves to an array ofTokenAllowance objects.

Example

```

Copy import{ getTokenAllowanceMulticall }from'@lifi/sdk';

constownerAddress='0x552008c0f6870c2f77e5cC1d2eb9bdff03e30Ea0'; consttokens=[{ token:{
address:'0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48', chainId:1, },
spenderAddress:'0x1231DEB6f5749EF6cE6943a275A1D3E7486F4EaE', }, { token:{
address:'0x6B175474E89094C44Da98b954EedeAC495271d0F', chainId:1, },
spenderAddress:'0x1231DEB6f5749EF6cE6943a275A1D3E7486F4EaE', },];

try{ constallowances=awaitgetTokenAllowanceMulticall(ownerAddress,tokens); console.log('Allowances:',allowances);
}catch(error) { console.error('Error:',error); }

```

setTokenAllowance

Sets the token allowance for a specific token and spender address.

Parameters

- request
- (ApproveTokenRequest): The approval request.
-
    - walletClient
-
    - (WalletClient): The wallet client used to send the transaction.
-
    - token
-
    - (BaseToken): The token for which to set the allowance.
-
    - spenderAddress
-
    - (string): The address of the spender.
-
    - amount
-
    - (bigint): The amount of tokens to approve.
-
    - infiniteApproval
-
```

    - (boolean, optional): If true, sets the approval to the maximum uint256 value.

Returns

A Promise that resolves to aHash representing the transaction hash orvoid if no transaction is needed (e.g., for native tokens).

Example

```

Copy import{ setTokenAllowance }from'@lifi/sdk';

constapprovalRequest={ walletClient:walletClient,// Viem wallet client token:{ address:'0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48', chainId:1, }, spenderAddress:'0x1231DEB6f5749EF6cE6943a275A1D3E7486F4EaE', amount:100000000n, };

try{ consttxHash=awaitsetTokenAllowance(approvalRequest); console.log('Transaction Hash:',txHash); }catch(error) { console.error('Error:',error); }

```

revokeTokenApproval

Revokes the token approval for a specific token and spender address.

Parameters

- request
- (RevokeApprovalRequest): The revoke request.
-
    - walletClient
-
    - (WalletClient): The wallet client used to send the transaction.
-
    - token
-
    - (BaseToken): The token for which to revoke the allowance.
-
    - spenderAddress
-
    - (string): The address of the spender.

Returns

A Promise that resolves to aHash representing the transaction hash orvoid if no transaction is needed (e.g., for native tokens).

Example

```

Copy import{ revokeTokenApproval }from'@lifi/sdk';

constrevokeRequest={ walletClient:walletClient,// Viem wallet client token:{ address:'0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48', chainId:1, }, spenderAddress:'0x1231DEB6f5749EF6cE6943a275A1D3E7486F4EaE', };

try{ consttxHash=awaitrevokeTokenApproval(revokeRequest); console.log('Transaction Hash:',txHash); }catch(error) { console.error('Error:',error); }

```

Testing Integration Last updated4 months ago