Query On-Chain Identifiers with ODIS

How to use ODIS to query the on-chain identifier given a phone number.

What is ODIS?

One of Celo's key features is the ability to associate a phone number to a Celo address. This provides a convenient payment experience for Celo users. To map a phone number to an address, the on-chain identifier for a given phone number must first be retrieved. With this identifier, the address can be looked up on-chain.

info ODIS requests are rate-limited based on transaction history and balance. Ensure the account that is performing the queries has a balance and has performed transactions on the network. If an out of quota error is hit, this indicates that more transactions need to be sent from the querying account. There are two methods for ODIS:

1. getPhoneNumberIdentifier
2.
    - Query and compute the identifier for a phone number
3. getContactMatches
4.
    - Find mutual connections between users

tip Seethis overview document for more details on ODIS.

Authentication

Both methods require authentication to the ODIS server, which can be performed by either the main wallet key or the data-encryption key (DEK) associated with the wallet key. This is managed byAuthSigner , which can be either aWalletKeySigner for a wallet key or anEncryptionKeySigner for the DEK. The DEK method is preferred, since it doesn't require the user to access the same key that manages their funds.You can learn more about DEK here.

You may use theEncryptionKeySigner for yourAuthSigner by passing in the raw private key:

const authSigner : AuthSigner =

{ authenticationMethod : OdisUtils . Query . AuthenticationMethod . ENCRYPTION_KEY , rawKey : privateDataKey , } ; Alternatively, you may use theWalletKeySigner by passing in a contractkit instance with the account unlocked:

const authSigner : AuthSigner =

{ authenticationMethod : OdisUtils . Query . AuthenticationMethod . WALLET_KEY , contractKit , } ;

Service Context

TheServiceContext object provides the ODIS endpoint URL and the ODIS public key (same as above).

const serviceContext : ServiceContext =

{ odisUrl , odisPubKey , } ; The ODIS endpoint URL for each environment can be found here:

Environment Key Alfajores Staging https://us-central1-celo-phone-number-privacy-stg.cloudfunctions.net Alfajores https://us-central1-celo-phone-number-privacy.cloudfunctions.net Mainnet https://us-central1-celo-pgpnp-mainnet.cloudfunctions.net The ODIS public key for each environment can be found here:

Environment Key Alfajores Staging
7FsWGsFnmVvRfMDpzz95Np76wf/1sPaK0Og9yiB+P8QbjiC8FV67NBans9hzZEkBaQMhiapzgMR6CkZIZPvgwQboAxl65JWRZecGe5V3XO4sdKeNemdAZ2TzQuWkuZoA
Alfajores
kPoRxWdEdZ/Nd3uQnp3FJFs54zuiS+ksqvOm9x8vY6KHPG8jrfqysvIRU0wtqYsBKA7SoAsICMBv8C/Fb2ZpDOqhSqvr/sZbZoHmQfvbqrzbtDIPvUIrHgRS0ydJCMsA
Mainnet
FvreHfLmhBjwxHxsxeyrcOLtSonC9j7K3WrS4QapYsQH6LdaDTaNGmnlQMfFY04Bp/K4wAvqQwO9/bqPVCKf8Ze8OZo8Frmog4JY4xAiwrsqOXxug11+htjEe1pj4uMA

Query phone number identifier

This call consumes quota. When the user runs out of quota, it's recommended to prompt the user to "purchase" more quota by sending a transaction to themselves. This method returns the pepper retrieved from the service as well as the the computed on-chain identifier that is generated using this pepper and the phone number.

BLS Blinding Client

It's important for user privacy that the ODIS servers don't have the ability to view the raw phone number. Before making the request, the library first blinds the phone number using a BLS library. This prevents the ODIS from being able to see the phone number but still makes the resulting signature recoverable to the original phone number. The blinding client is written inRust and compiled to Web Assembly, which is not compatible with React native. If you choose not to pass in aBLSBlindingClient it will default to the Web Assembly version. You may create aReactBlindingClient by calling the constructor with the ODIS public key:

const blsBlindingClient =

new

ReactBlsBlindingClient (odisPubKey) ; Or use theWasmBlsBlindingClient if your runtime environment supports Web Assembly:

const blsBlindingClient =

new

WasmBlsBlindingClient (odisPubKey) ; Now you're ready to get the phone number identifier.OdisUtils.PhoneNumberIdentifier.getPhoneNumberIdentifiedocumentation can be found here .

The response will bean object with the original phone number, the on-chain identifier (phoneHash), and the phone number's pepper.

You can view an example of this call inour mobile project here .

Matchmaking

Instead of querying for all the user's contact's peppers and consuming the user's quota, it's recommended to only query the pepper before it's actually used (ex. just before sending funds). However, sometimes it's helpful to let your users know that they have contacts already using the Celo network. To do this, you can make use of the matchmaking interface. Given two phone numbers, it will let you know whether the other party has also registered on the Celo network with this identifier.OdisUtils.Matchmaking.getContactMatches documentation can be found here .

The response will be a subset of the inpute164NumberContacts that are matched by the matchmaking service.

You can view an example of this call in our mobile project here . Edit this page Previous Deploy and Interact with Contracts (Remotely)Next react-celo