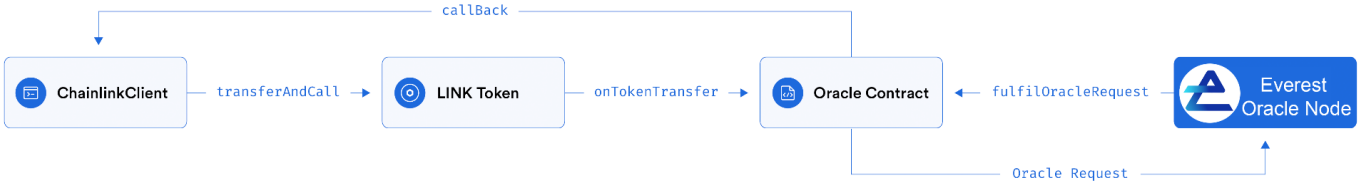
The "How To" Guide to Use Everest Identity Oracle Services through Chainlink Direct Requests

The Chainlink Direct Request Flow to Everest Oracle Node:



Everest Direct Requests:

Type of Request	Example Request:	Example Responses:
Human & Unique	Is 0xb794f5ea0ba39494ce839613fffba74279579268 Human and Unique?	**"Yes"** or **"Unknown"**
KYCed	Is 0xb794f5ea0ba39494ce839613fffba74279579268 KYCed?	**"Yes on 4 November 2022"** or **"Unknown"**

Table of Contents:

1. Everest Address Check Protocol

Smart contract developers will have a full Web 3.0 infrastructure stack including blockchains for on-chain logic and state changes, Chainlink oracles for off-chain communication and computation, and Everest for unique human identity, account creation/verification, KYC/AML, and KYB organisational registration.

By integrating Chainlink into the Everest ecosystem, several innovative bidirectional smart contract capabilities are now possible. This includes leveraging Chainlink to bridge cryptographically-proven biometric identity accessed via Everest's Everchain to external smart contract systems as a means of triggering their on-chain applications, as well as using Chainlink oracles to enable smart contracts to trigger Everest KYC/KYB capabilities. Since Everest will use Chainlink infrastructure, applications will be able to call the Everest oracle, to determine if an associated wallet is in one of four states: unknown, Human and Unique, Identity Verified, and KYC Status checked. Organisations are in one three states: unknown, Unverified, and Verified (KYB). KYCed and KYBed organisations can use fiat or crypto-in/out facilities that are provided by Everest. Users will ALWAYS maintain control over their data, what gets shared, with whom and how; and any credential sharing will be accomplished privately and securely.

This bidirectional data partnership positions Chainlink as the most frictionless solution for any smart contract developer wanting to perform KYC, human & unique identity verification between Everest and external systems. Chainlink has a large collection of security reviewed, Sybil-resistant node operators which can be easily composed into decentralised oracle networks to guarantee strong uptime and tamper-resistance around oracle services. This integration greatly expands the scope of applications developers can create due to Chainlink's blockchain and API agnostic ecosystem.

Chainlink opens up bi-directional communication capabilities for Everest, meaning it can receive external inputs that trigger Everest's identity account creation/verification or fetch a user/business's KYC/KYB status. This is enabled through the development of custom Chainlink External Adapters, which can read and write data on Everchain.

Initially, an Everest external adapter allows Chainlink oracles to read a user (KYC) or organisation's (KYB) verification status within the Everest ecosystem, which can then be combined with other external adapters to write that data on other networks. There are also many other possibilities here, including account creation from other blockchain networks directly onto Everchain.

Initial use cases include:

1. Everest offers human and unique status oracle through Chainlink by enabling the distribution of credentials - which allow the Chainlink Oracle subscriber to query if this is a known individual. For example, prevent a user from deduplication within an external ecosystem.
2. Everest offers Identity Verified KYCed status oracle through Chainlink by enabling the distribution of credentials which allow the Chainlink Oracle subscriber to query if this individual's KYC status has been recently checked. This allows organisations to scale up quickly reducing operational costs while still maintaining KYC/AML compliance.
3. Everest offers KYCed status oracle through Chainlink by enabling the distribution of credentials which allow the Chainlink Oracle subscriber to query if this individual's KYC status has been recently checked.
4. Everest offers organisational registration and KYB status oracle through Chainlink by enabling the distribution of credentials which allow the Chainlink Oracle subscriber to query if this organisation’s KYB status has been recently checked. Most B2B companies need to meet mandatory legal compliance by performing due diligence in identifying the legal representative(s) and their connection to the company. Everest takes this tedious process and greatly reduces costs, time and administrative procedures.
5. Everest and Chainlink co-create a "monetize my personal information" marketplace. It puts the user in control of their private permissioned identity, and allows them to surface pieces of information about themselves for incentives if they choose so.
6. Everest and Chainlink co-create a creditworthiness oracle offering.

2. Data Request and Response Formats

Key Functionality:

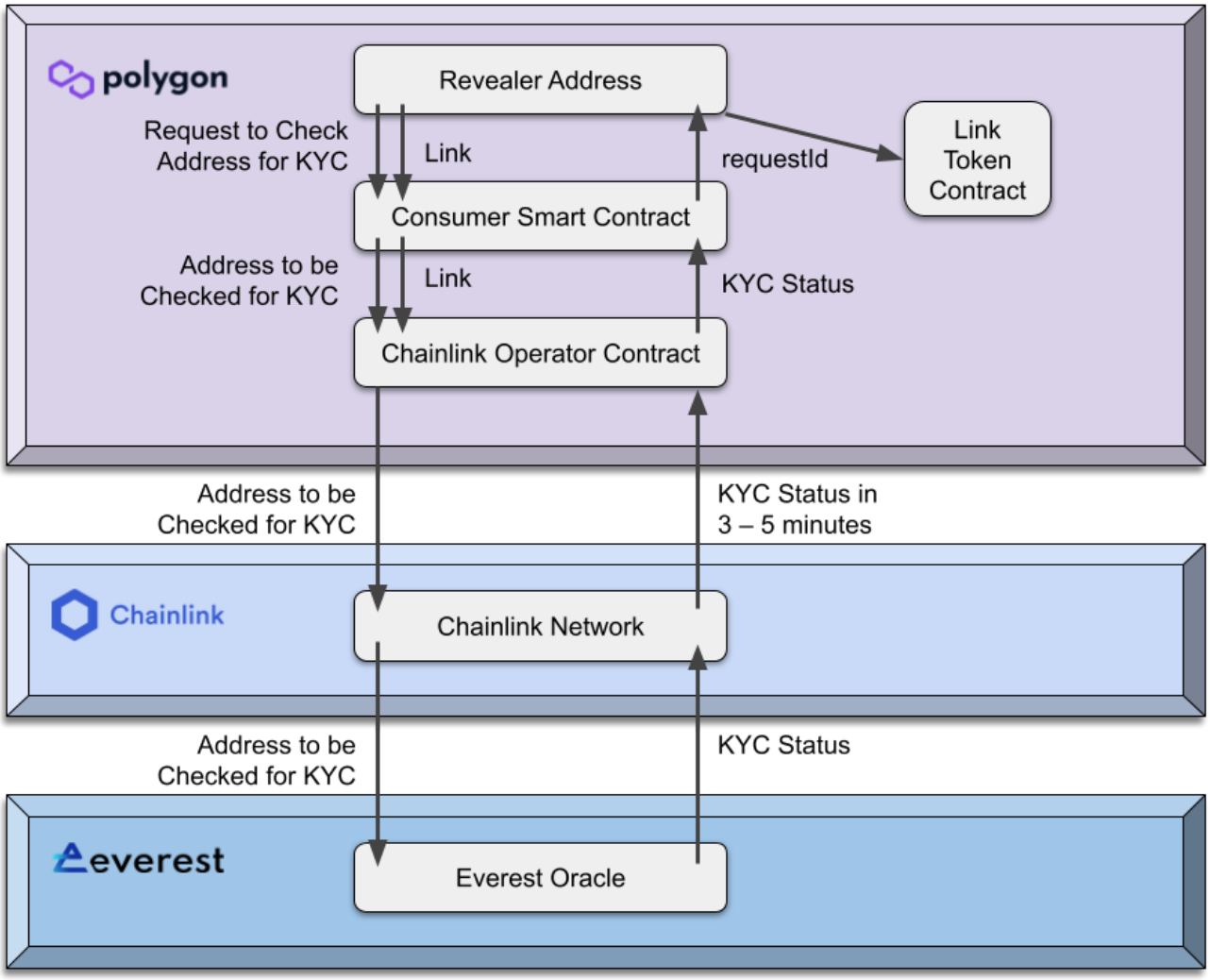
- Consumer Smart Contract on external network (production - Polygon) is able to make “get address status” request by passing an address and getting back requestId;
- Consumer Smart Contract is able to get response by requestId;
- The response contains data about address status (Human&Unique or not, date of KYCed if available);
- Data about all users on Everest platform will be used to perform the check (combine the status response);
- Communications between networks are provided by Chainlink;
- Chainlink request from Polygon network will be relayed to Everest platform and answered after LINK token payment has been received.

Consumer Smart Contract Repository:
https://github.com/EverID/everest-chainlink-consumer/blob/master/contracts/EverestConsumer.sol

Two Steps KYC Status Request

1) Status Request

- Allow Consumer Smart Contract (EverestConsumer) contract spend your LINK tokens:
  LinkTokenInterface.approve(
  address spender, - everest consumer contract
  uint256 value - more than "payment"
)
- Request the status (this method also makes transferFrom):
  EverestConsumer.requestStatus(
  address _revealee - desired address
)
- Retrieve your latest requestId calling this method:
  EverestConsumer.getLatestSentRequestId()
- Wait about 2-3 minutes before the request will be fulfilled.


polygon
Revealer Address
Request to Check
Address for KYC
Link
requestId
Link
Token
Contract
Consumer Smart Contract
Address to be
Checked for KYC
Link
KYC Status
Chainlink Operator Contract
Address to be
Checked for KYC
KYC Status in
3 – 5 minutes
Chainlink
Chainlink Network
Address to be
Checked for KYC
KYC Status
everest
Everest Oracle

2) KYC Request

- You can get any request by requestId or the latest fulfilled request by the address using the following methods:

  getRequest(bytes32 _requestId)

  getLatestFulfilledRequest(address _revealee)

- If *getRequest* method returns *isFulfilled=false* for 5 minutes, you can cancel your request and return funds using:

  cancelRequest(bytes32 _requestId)

KYC Request Structure:

```
{
   bool isFullfilled; // 1 byte - slot 0
   bool isCanceled; // 1 byte - slot 0
   bool isHumanAndUnique; // 1 byte - slot 0
   bool isKYCUser; // 1 byte - slot 0
   address revealer; // 20 bytes - slot 0
   address revealee; // 20 bytes - slot 1
// 'kycTimestamp' is zero if the status is not 'KYCUser'
// otherwise it is an epoch timestamp that represents the KYC date
uint40 kycTimestamp; // 5 bytes - slot 1
   // expiration = block.timestamp while 'requestStatus' + 5 minutes
   // if 'isFullfilled' and 'isCanceled' are false by this time -
   // the request owner can cancel its request
   // using 'cancelRequest' and return paid LINK tokens
uint40 expiration; //
}
```

3. Integration Guides

Preparation Steps

1. Import LinkToken Interface to your smart contract

```
import "@chainlink/contracts/src/v0.8/interfaces/LinkTokenInterface.sol";
```

2. Init object

```
erc20Link = LinkTokenInterface(linkAddr);
```

3. Make Approve

```
erc20Link.approve(consAddr, type(uint256).max);
```

4. Make sure you have at least 0.1 LINK - for one request

Main Steps

1. Import IEverestConsumer to your smart contract

```
import "everest-consumer/interfaces/IEverestConsumer.sol";
```

2. Initialise object

```
consumer = IEverestConsumer(consAddr);
```

3. Make Request

```
consumer.requestStatus(revealeeAddress)
```

4. Wait some time, and check the result

```
result = consumer.getLatestFulfilledRequest(revealeeAddress)
```

Network Details

Polygon (Matic) Mainnet

LINK Token Address: 0xb0897686c545045aFc77CF20eC7A532E3120E0F1
Operator Address: 0x97b6Df5808b7f46Ee2C0e482E1B785CE3A2BC8BF
Consumer Smart Contract: 0xC1AfF12173B38aE44feDF453Af7A57AFF3cFd3f0
Payment Amount: 0.1 LINK
JobID: 827352c4d8684571b4605f9022853ddf
JobID as bytes32:
0x3832373335326334643836383435373162343630356639303232383533646466

Ethereum Goerli Testnet

LINK Token Address: 0x326C977E6efc84E512bB9C30f76E30c160eD06FB
Operator Address: 0xB9756312523826A566e222a34793E414A81c88E1
Consumer Smart Contract: 0xd6c576B4f6Ab3d70b49FA2a1F73711943f3a14f2
Payment Amount: 0.1 LINK
JobID: 14f849816fac426abda2992cbf47d2cd JobID as bytes32:
0x3134663834393831366661633432366162646132393932636266343764326364

4. How to Make a KYC Request

We will go through the KYC request procedure with smart contracts deployed on Goerli Testnet.

1) Initial request with Address 0x652c3c775A82fEc8D176BEaEB1e259DD5b0c8526 to be checked for KYC from 0xb1080E639CC542C9BfbC1e2fffac075Fea848287 performed with Consumer Smart Contract

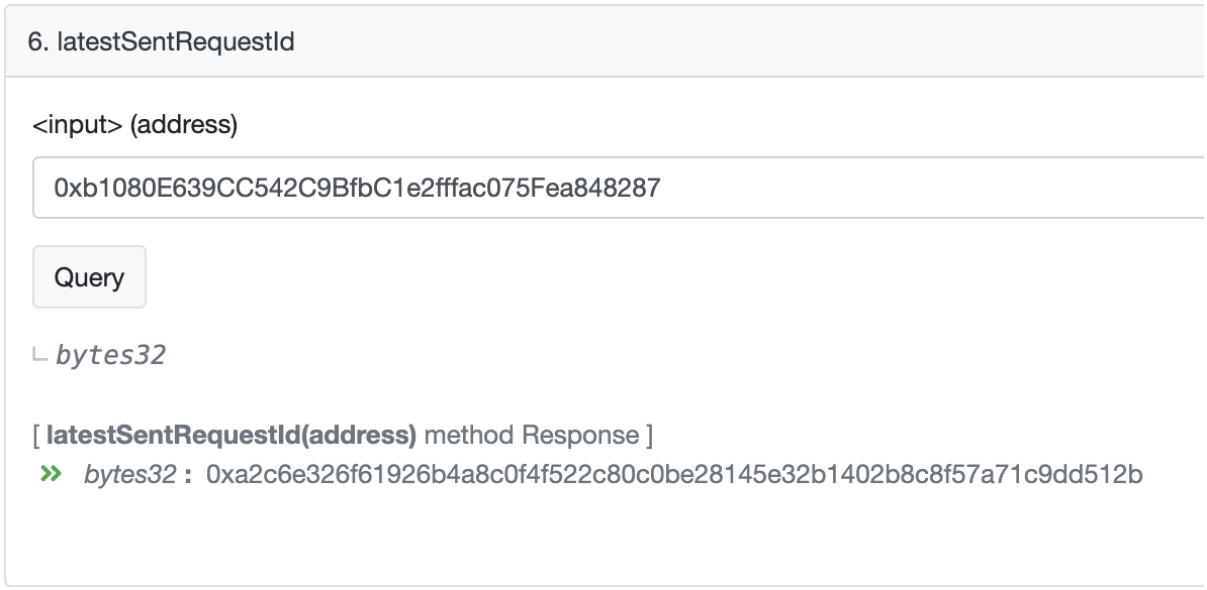
Link to Consumer Smart Contract:
https://goerli.etherscan.io/address/0xd6c576b4f6ab3d70b49fa2a1f73711943f3a14f2#writeContract

The following transaction is performed:
https://goerli.etherscan.io/tx/0x228e88515a1bf2aa51786ab23844bc64687f98aa13e60e4b8f22a3fb520ecdf1

Link to Chainlink Operator Contract:
https://goerli.etherscan.io/address/0xb9756312523826a566e222a34793e414a81c88e1

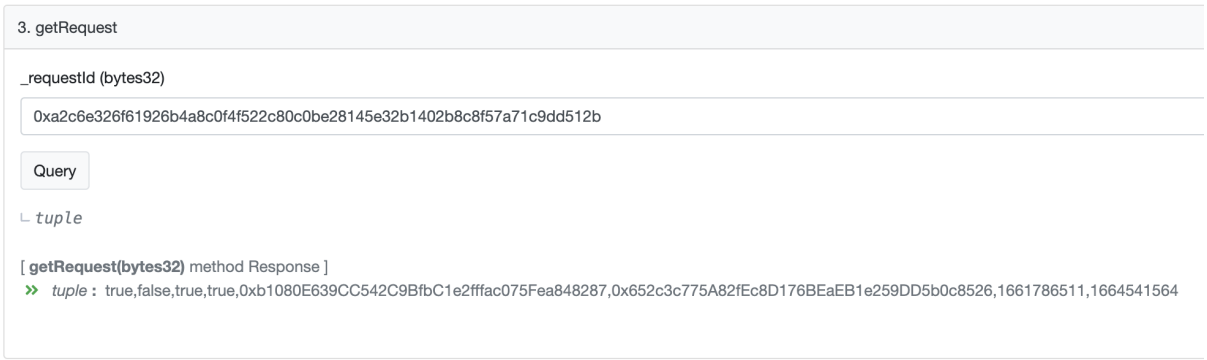
Consumer Smart Contract sends the following transaction to Chainlink Operator Contract:
https://goerli.etherscan.io/tx/0x47055719d33398c1c13a2a0421048e53f4c79c6d4228e70c3b53b4b0dcc0ce8d

2) In response to initiated request Consumer Smart Contract provide us with requestID: 0xa2c6e326f61926b4a8c0f4f522c80c0be28145e32b1402b8c8f57a71c9dd512b



Link to Consumer Smart Contract:
https://goerli.etherscan.io/address/0xd6c576b4f6ab3d70b49fa2a1f73711943f3a14f2#readContract

3) After period of 3-5 minutes when Everest chain KYC request is performed we can obtain KYC data from Consumer Smart Contract

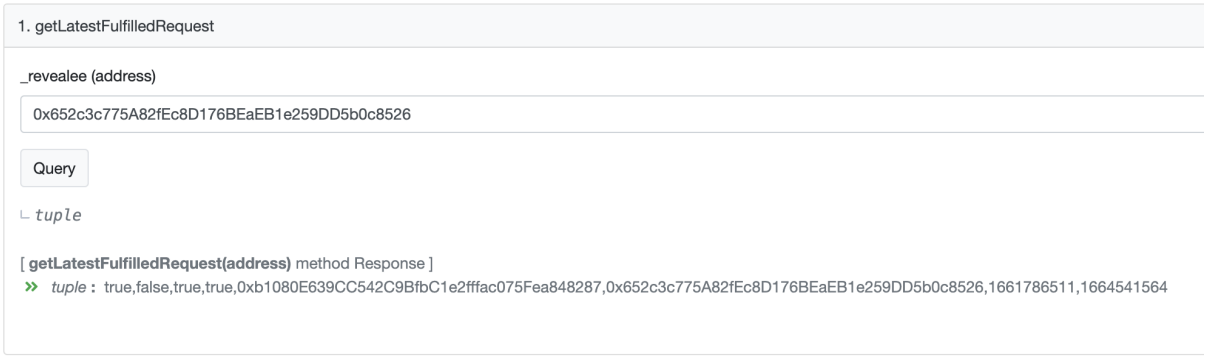


Obtained from Consumer Smart Contract KYC Status:
true,false,true,true,0xb1080E639CC542C9BfbC1e2fffac075Fea848287,0x652c3c775A82fEc8D176BEaEB1e259DD5b0c8526,1661786511,1664541564

We interpret according to KYC Request Structure:

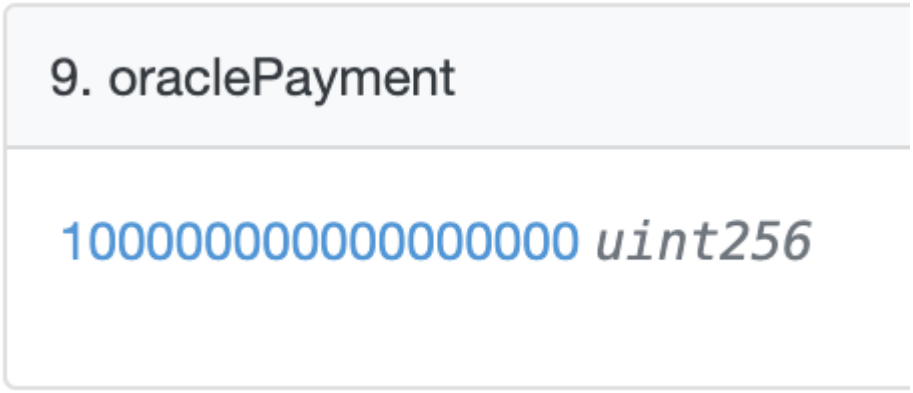
{
bool isFullfilled - true
bool isCanceled - false
bool isHumanAndUnique - true
bool isKYCUser - true
address revealer - 0xb1080E639CC542C9BfbC1e2fffac075Fea848287
address revealee - 0x652c3c775A82fEc8D176BEaEB1e259DD5b0c8526
uint40 kycTimestamp - 1661786511 / GMT: Monday, August 29, 2022 3:21:51 PM
uint40 expiration - 1664541564 / Friday, September 30, 2022 12:39:24 PM
}

We can get the same KYC data by providing revealee address
0x652c3c775A82fEc8D176BEaEB1e259DD5b0c8526



Pricing for KYC Requests

Actual price for KYC requests could be obtained from Consumer Smart Contract



https://goerli.etherscan.io/address/0xd6c576b4f6ab3d70b49fa2a1f73711943f3a14f2#readContract

100000000000000000 is equal to 0.1 LINK tokens per requestId

Useful Links

Everest ID Oracle Overview
https://gist.github.com/ericjaurena/e29060362f311158dfccff94c95dd9c8