Smart contract on the Solana blockchain that interacts with the Interledger Protocol (ILP)

An example smart contract on the Solana blockchain that interacts with the Interledger Protocol (ILP) verifies hashes of data from the DHT table and matches them with data obtained from the CID in Helium.

If the hash value of the data from the DHT table and the CID match, then a reward is awarded in Solana tokens:

```
use anchor_lang::prelude::*;
use anchor_lang::solana_program::system_program; use
anchor_spl::token::{self, Mint, Token, TokenAccount}; use
ilp_sdk::ilp_client::IlpClient;
use helium_sdk::helium_client::HeliumClient;

declare_id!("your_program_id_here");

#[program]
pub mod data_verification {
    use super::*;

    pub fn verify_and_reward(ctx: Context<VerifyAndReward>, dht_hash: String, cid: String) ->
ProgramResult {
        let ilp_client = IlpClient::new("ilp_node_url");
        let helium_client = HeliumClient::new("helium_api_url");

        // Getting data from the DHT table by hash let
        dht_data = ilp_client.get_dht_data(&dht_hash)?;

        // Receive data from Helium by CID
        let helium_data = helium_client.get_data_by_cid(&cid)?;

        // Data comparison
        if dht_data == helium_data {
            // Accrual of rewards in Solana tokens let
            reward_amount = 10_000_000; // 10 SOL let
            cpi_accounts = token::Transfer {
                from: ctx.accounts.user_token_account.to_account_info(), to:
                ctx.accounts.reward_token_account.to_account_info(),
                authority: ctx.accounts.user.to_account_info(),
            };
            let cpi_token_program = ctx.accounts.token_program.to_account_info(); let
            cpi_ctx = CpiContext::new(cpi_token_program, cpi_accounts);
            token::transfer(cpi_ctx, reward_amount)?;
        }
        Ok(())
    }
}

# [derive(Accounts)]
pub struct VerifyAndReward<'info> {
    # [account(mut)]
    pub user: Signer<'info>,
    # [account(mut)]
    pub user_token_account: Account<'info, TokenAccount>,
    # [account(mut)]
    pub reward_token_account: Account<'info, TokenAccount>,
    pub token_program: Program<'info, Token>,
    pub system_program: Program<'info, System>,
}
```

1. The smart contract declares the verify_and_reward function, which receives the hash of the data from the DHT and CID table from Helium.

2.Clients are created to interact with the Interledger Protocol (ILP) and Helium.

3.Data is retrieved from the DHT table by hashing using the ILP client.

4.Data is retrieved from Helium using CID using the Helium client.

5. The obtained data is compared.
6.If they match, a reward is awarded in Solana tokens.

To calculate the reward, we use the token::transfer command from the anchor_spl library, which transfers the specified number of tokens from the user account to the reward account.

The VerifyAndReward structure defines the accounts required to execute the verify_and_reward function:
user:
The user account calling the function (must be authorized). user_token_account: user token account from which bonus tokens will be debited.

reward_token_account: The token account to which bonus tokens will be credited.
token_program: Solana token program.
system program: Solana system program.

Note that this example assumes that the ilp_sdk and helium_sdk libraries will interoperate with the Interledger and Helium protocols, respectively.
We need to implement these libraries or find existing implementations to use them in the smart contract.

You also need to remember to replace your_program_id_here with your real program identifier.

Before deploying the smart contract, let's make sure that we have the Solana development environment configured and the necessary dependencies such as Rust and Anchor Framework installed.

Once the smart contract is deployed, we can call the verify_and_reward function, passing the hash of the data from the DHT table and the CID from Helium. The smart contract will check the data and issue a reward in Solana tokens if the data matches.

It is necessary to rewrite this contract in such a way that it can be connected by Helium networks users to the Solana blockchain.