# Custom Deposit Address Fix - Summary

## Problem Description

### Initial Issue

1. In the admin panel, when creating a wallet, there was a text field "Type network" where "TRX" was manually typed
2. On the deposit page, "USDT" text was shown instead of the real custom address `TK5F8mcAHAcgetuhr25ypfQUEql`
3. The problem was that the network field in the admin form was a text input (allowing any text), while the deposit page searched using different logic

## Root Cause Analysis

The mismatch occurred because:

- **Admin Panel**: Network field was a text input where users could type anything (e.g., "TRX")
- **Deposit Page**: System expected specific network identifiers like "TRC20", "ERC20", etc.
- **Matching Logic**: The search was looking for exact matches between `selectedDepositMethod` (e.g., "TRC20") and the stored `network` field (e.g., "TRX")

## Solution Implemented

### 1. Changed Admin Panel - Network Field to Dropdown

**File**: `src/components/elements/base/datatable/FormModal/AddressWalletsPairFields/AddressWalletsPairFields.tsx`

**Changes:**

- **Before**: Text input field for network
- **After**: Dynamic dropdown that loads available networks based on selected currency

**Key Features:**

```
// Added state to store networks for each currency
const [networkOptions, setNetworkOptions] = useState<Record<string, any[]>>({});
const [loadingNetworks, setLoadingNetworks] = useState<Record<string, boolean>>({});

// Function to fetch networks when currency is selected
const fetchNetworksForCurrency = useCallback(async (currency: string, index: number) => {
  const { data, error } = await $fetch({
    url: `/api/finance/currency/SPOT/${currency}?action=deposit`,
    silent: true,
  });

  if (!error && data) {
    const networks = Array.isArray(data) ? data.map(network => ({
      value: network.chain,
      label: network.chain,
      network: network.chain,
    })) : [];
    setNetworkOptions(prev => ({ ...prev, [currency]: networks }));
  }
}, []);
```

**Network Field Now Shows:**

- TRC20
- ERC20
- BEP20
- And other available networks for the selected currency

## 2. Enhanced Deposit Page Matching Logic

**File**: `src/components/pages/user/wallet/deposit/DepositAddress/DepositAddress.tsx`

**Improved Matching Algorithm:**

```
useEffect(() => {
  console.log("=== [DepositAddress] Custom Address Matching Debug ===");

  if (Array.isArray(profile?.customAddressWalletsPairFields)) {
    // Step 1: Try to find exact match (currency + network)
    let currentAddress = profile.customAddressWalletsPairFields.find(
      (obj) => {
        const currencyMatch = obj.currency === selectedCurrency;
        const networkMatch = obj.network === selectedDepositMethod;
        console.log(`Checking: currency=${obj.currency} (match:
${currencyMatch}), network=${obj.network} (match: ${networkMatch})`);
        return currencyMatch && networkMatch;
      }
    );

    // Step 2: Fallback to currency-only match
    if (!currentAddress) {
      currentAddress = profile.customAddressWalletsPairFields.find(
        (obj) => obj.currency === selectedCurrency
      );
    }
    // Step 3: Use system deposit address as last resort
    if (!currentAddress) {
      currentAddress = depositAddress;
    }

    setUserDepositAddress(currentAddress);
  }
}, [profile, selectedCurrency, selectedDepositMethod, depositAddress]);
```

### 3. Updated Backend Structure

**File**: `backend/api/admin/crm/user/structure.get.ts`

```
const customAddressWalletsPairFields = {
  type: "customAddressWalletsPairFields",
  label: "custom Address Wallets Pair Fields",
  name: "customAddressWalletsPairFields",
  placeholder: "Enter custom fields for this wallet",
  fields: {
    address: walletAddressConditions?.wallets,
    currency: currencyConditions?.SPOT,
    // network will be dynamically loaded based on selected currency
  },
};
```

## Debug Logging Added

Comprehensive console logging for troubleshooting:

### Admin Panel Logs:

```
[AddressWalletsPairFields] Fetching networks for currency: USDT
[AddressWalletsPairFields] Networks fetched for USDT: [...]
```

### Deposit Page Logs:

```
=== [DepositAddress] Custom Address Matching Debug ===
[DepositAddress] selectedCurrency: USDT
[DepositAddress] selectedDepositMethod: TRC20
[DepositAddress] profile.customAddressWalletsPairFields: [...]
[DepositAddress] Checking: currency=USDT (match: true), network=TRC20 (match: true), address=TK5F8mcAHAcgetuhr25ypfQUEql
[DepositAddress] ✅ Found exact match (currency + network): {...}
[DepositAddress] Final address to use: {...}
=== [DepositAddress] End Debug ===
```

## How It Works Now

### Admin Panel Flow:

1. Admin selects currency (e.g., "USDT") from dropdown
2. System automatically fetches available networks for USDT via API: `/api/finance/currency/SPOT/USDT?action=deposit`
3. Network dropdown populates with options: TRC20, ERC20, BEP20, etc.
4. Admin selects network (e.g., "TRC20")
5. Admin enters or selects custom wallet address: `TK5F8mcAHAcgetuhr25ypfQUEql`
6. Data saved to user profile as:

```
{
  "currency": "USDT",
  "network": "TRC20",
  "address": "TK5F8mcAHAcgetuhr25ypfQUEql"
}
```

### User Deposit Flow:

1. User selects currency: USDT
2. User selects network: TRC20
3. System searches custom addresses:
    - First: Exact match (currency=USDT AND network=TRC20) ✅
    - Fallback: Currency only match (currency=USDT)
    - Last resort: System deposit address
4. Shows custom address: `TK5F8mcAHAcgetuhr25ypfQUEql`

## Expected Result

When testing with:

- **Currency**: USDT
- **Network**: TRC20
- **Custom Address**: `TK5F8mcAHAcgetuhr25ypfQUEql`

The deposit page will now correctly display the custom address instead of "USDT".

## Testing Checklist

- □ Open Admin Panel → Users Management
- □ Edit a user and add custom wallet:
    - □ Select Currency: USDT
    - □ Wait for network dropdown to load
    - □ Select Network: TRC20
    - □ Enter Address: `TK5F8mcAHAcgetuhr25ypfQUEql`

    - □ Save
- □ Login as that user
- □ Go to Deposit page
- □ Select USDT currency
- □ Select TRC20 network
- □ Verify custom address `TK5F8mcAHAcgetuhr25ypfQUEql` is displayed
- □ Check browser console for debug logs

## Files Modified

1. `src/components/elements/base/datatable/FormModal/AddressWalletsPairFields/AddressWalletsPairFields.tsx`
    - Changed network input from text to dropdown
    - Added dynamic network loading based on currency
    - Added loading states and error handling
2. `src/components/pages/user/wallet/deposit/DepositAddress/DepositAddress.tsx`
    - Enhanced matching logic with detailed logging
    - Improved fallback mechanism
    - Added comprehensive debug console logs
3. `backend/api/admin/crm/user/structure.get.ts`
    - Updated field structure documentation
    - Removed static network field

## API Endpoints Used

- **GET** `/api/finance/currency/SPOT/{currency}?action=deposit`
    - Returns available deposit networks for a currency
    - Example response for USDT:

```
[
  {
    "id": "...",
    "chain": "TRC20",
    "fee": 1,
    "precision": {...},
    "limits": {...}
  },
  {
    "id": "...",
    "chain": "ERC20",
    "fee": 5,
    "precision": {...},
    "limits": {...}
  }
]
```

## Benefits

1. ☑ **Data Consistency**: Network values are now standardized (TRC20, ERC20, etc.)
2. ☑ **User Experience**: Admin can't enter invalid network names
3. ☑ **Debugging**: Comprehensive logs for troubleshooting
4. ☑ **Matching Accuracy**: Exact match logic ensures correct address display
5. ☑ **Maintainability**: Centralized network data from API

## Potential Issues & Solutions

### Issue: Networks not loading

**Solution**: Check console for errors, verify API endpoint is accessible

### Issue: Old data with text network values

**Solution**: Admin needs to re-edit and select proper network from dropdown

### Issue: Custom address still not showing

**Solution**: Check console logs to see matching process, verify currency and network match exactly

## Migration Notes

For existing custom wallet data with text network values (e.g., "TRX" instead of "TRC20"):

1. Admin should edit the user's custom wallet configuration
2. Select the currency again to trigger network loading
3. Select the correct network from the dropdown
4. Save the updated configuration

This ensures all data uses standardized network identifiers.