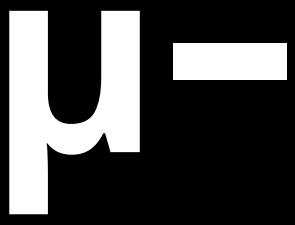
DAPPS

Muon Pay SDK Documentation



Reach out to rxddapps for integration into your website or service.
Contact via email : contact@rxddapps.com
Website: rxddapps.com

rxddapps.com

DAPPS

Why "Muon Pay SDK"?

The name Muon Pay (μ-) is inspired by the muon, a fascinating subatomic particle in physics. Similar to an electron but much heavier, the muon moves at incredible speeds and interacts with matter while experiencing minimal energy loss—a perfect analogy for low-fee transactions on the Radiant blockchain. Just as muons are single-use and short-lived, the deposit addresses in Muon Pay are generated per transaction, ensuring security and efficiency. Additionally, like how a muon is a fundamental, indivisible particle, Radiant's transactions are measured in photons, the smallest unit of RXD.

By adopting the Muon Pay SDK, merchants can leverage the power of fast, low-cost payments on Radiant, providing a seamless and reliable experience for digital commerce.

μ⁻

rxddapps.com

DAPPS

Introduction:

The RXDDApps SDK allows you to send and receive payments on the Radiant Blockchain using RXD tokens (or Photons). This SDK simplifies integration for handling transactions and managing payments in your project.

How the SDK Works:

1. **Photons and RXD Tokens:**
   - Payments are made using Photons, which are the smallest unit of RXD tokens.
   - 1 Photon = 0.00000001 RXD.

2. **Key Features:**
   - Send and receive payments using Radiant addresses.
   - Monitor balances and transaction statuses.
   - Create and manage deposit addresses for payments.
   - Receive postbacks for payment confirmations.

DAPPS

API Endpoints and Methods:

1. **Generate a New Deposit Address:**

- Endpoint: */api/user/transaction/new/address*
- Method: POST

```
{
    "userid": 12345,
    "amountNeeded": 1000000 // in photons
}
```

- userid: Unique identifier for the user.
- amountNeeded: Minimum amount to mark the transaction as paid (in Photons).

Possible Responses:

- **Success (Address Retrieved):**

DAPPS

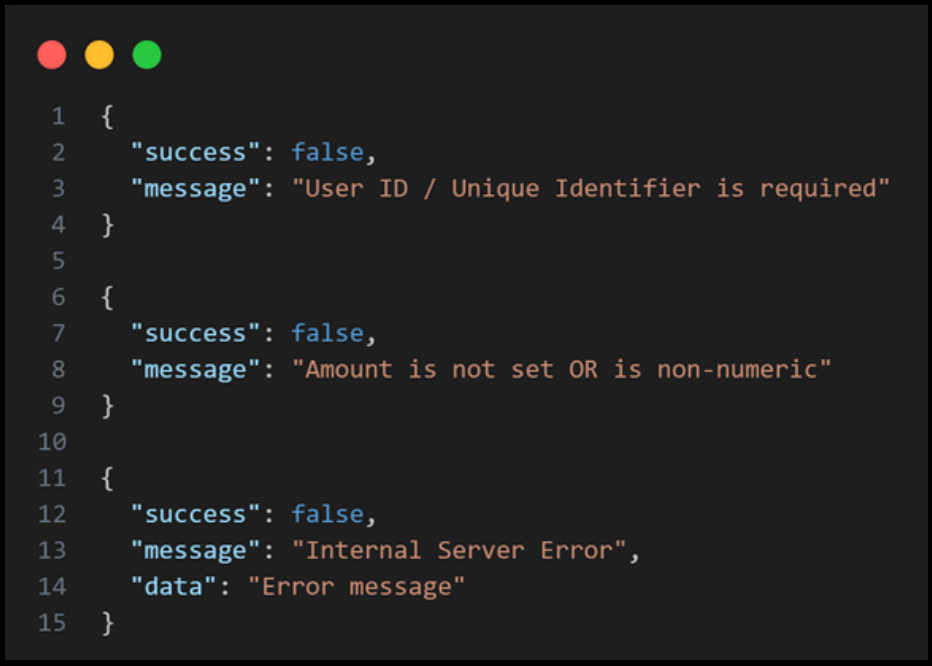
```
{
    "success": true,
    "message": "Address retrieved successfully",
    "data": {
      "addressId": 1,
      "userid": 12345,
      "amount": 1000000,
      "fee": 10000,
      "address": "generated_address_here",
      "time": "timestamp_here"
    }
  }
```

- **Success (New Address Generated):**

```
{
    "success": true,
    "message": "Address generated successfully",
    "data": {
      "addressId": 2,
      "userid": 12345,
      "amount": 1000000,
      "fee": 10000,
      "address": "new_generated_address_here",
      "time": "timestamp_here"
    }
  }
```

DAPPS

- **Error Responses:**

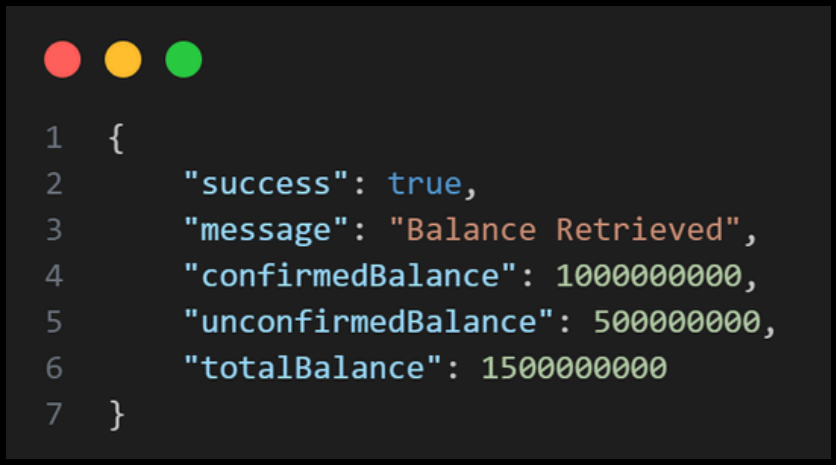


```
{
  "success": false,
  "message": "User ID / Unique Identifier is required"
}

{
  "success": false,
  "message": "Amount is not set OR is non-numeric"
}

{
  "success": false,
  "message": "Internal Server Error",
  "data": "Error message"
}
```

2. Check Your Balance:

- Endpoint: */api/user/transaction/balance*
- Method: POST
- Request Body: No input required. (Uses *DEFAULT_ADDRESS* from ***.env*** file)

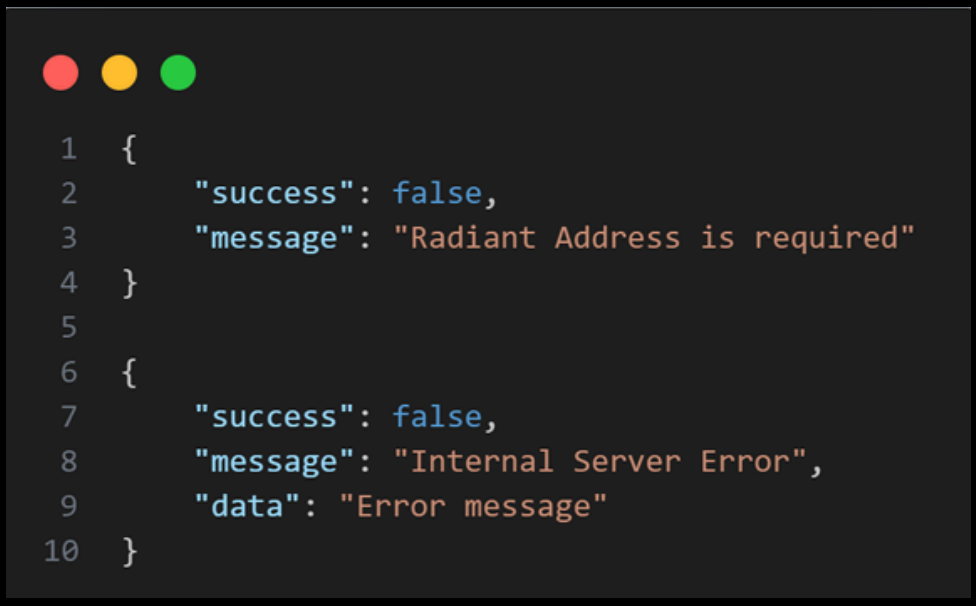
- **Possible Responses:**
  - **Success:**



```
{
    "success": true,
    "message": "Balance Retrieved",
    "confirmedBalance": 1000000000,
    "unconfirmedBalance": 500000000,
    "totalBalance": 1500000000
}
```

- confirmedBalance: Confirmed balance in Photons.
- unconfirmedBalance: Unconfirmed balance in Photons.
- totalBalance: Total balance in Photons.

- **Error Responces:**



```
{
    "success": false,
    "message": "Radiant Address is required"
}

{
    "success": false,
    "message": "Internal Server Error",
    "data": "Error message"
}
```

3. Check for Deposits

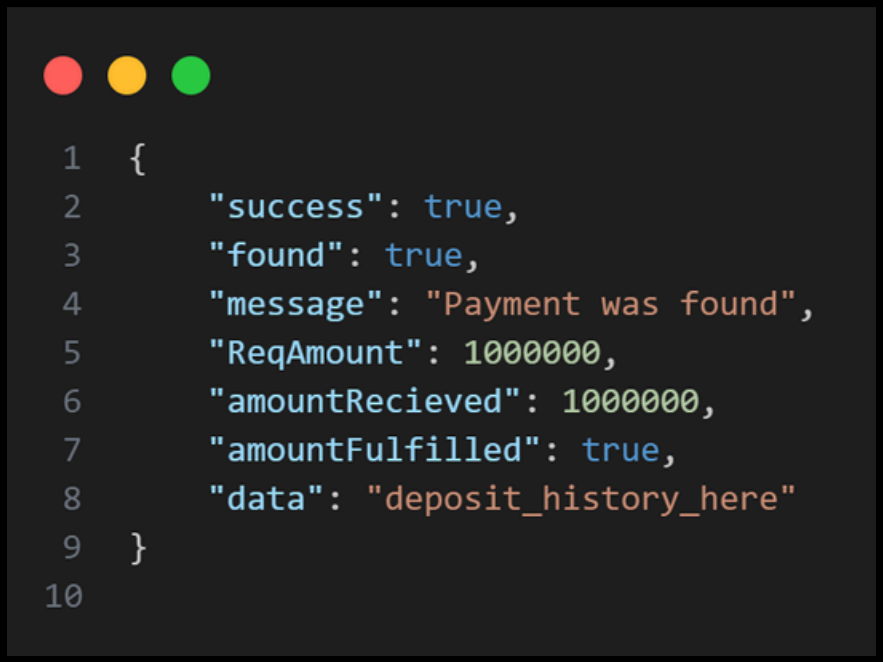
- **Endpoint:** */api/user/transaction/check/deposit*
- **Method: POST**
- **Request Body:**

DAPPS

```
{
    "address": "deposit_address_here"
}

```

- address: The deposit address to check.

- **Possible Responses:**
  - **Success (Payment Found):**



```
{
    "success": true,
    "found": true,
    "message": "Payment was found",
    "ReqAmount": 1000000,
    "amountRecieved": 1000000,
    "amountFulfilled": true,
    "data": "deposit_history_here"
}

```

DAPPS

- **Success (No Payment Found):**

```
{
    "success": true,
    "found": false,
    "message": "No Payments found for this address",
    "data": []
}
```

- **Error Responses:**

```
{
    "success": false,
    "message": "Address is required"
}
{
    "success": false,
    "message": "Address not found or already used"
}
{
    "success": false,
    "message": "Internal Server Error",
    "data": "Error message"
}
```

DAPPS

4. Broadcast a Transaction

- Endpoint: */api/user/transaction/broadcast*
- Method: POST
- Request Body:

```
{
    "myAddress": "sender_address_here",
    "toAddress": "recipient_address_here",
    "amount": 1000000
}
```

- myAddress: Sender's Radiant address.
- toAddress: Recipient's Radiant address.
- amount: Amount to send in Photons (e.g., 1 RXD = 100000000 Photons).

Note: The Private key is to be stored inside .env file

- **Possible Responses:**
  - **Success:**

```
{
    "success": true,
    "message": "Transaction broadcasted successfully",
    "data": { "txId": "transaction_id_here" }
}
```

DAPPS

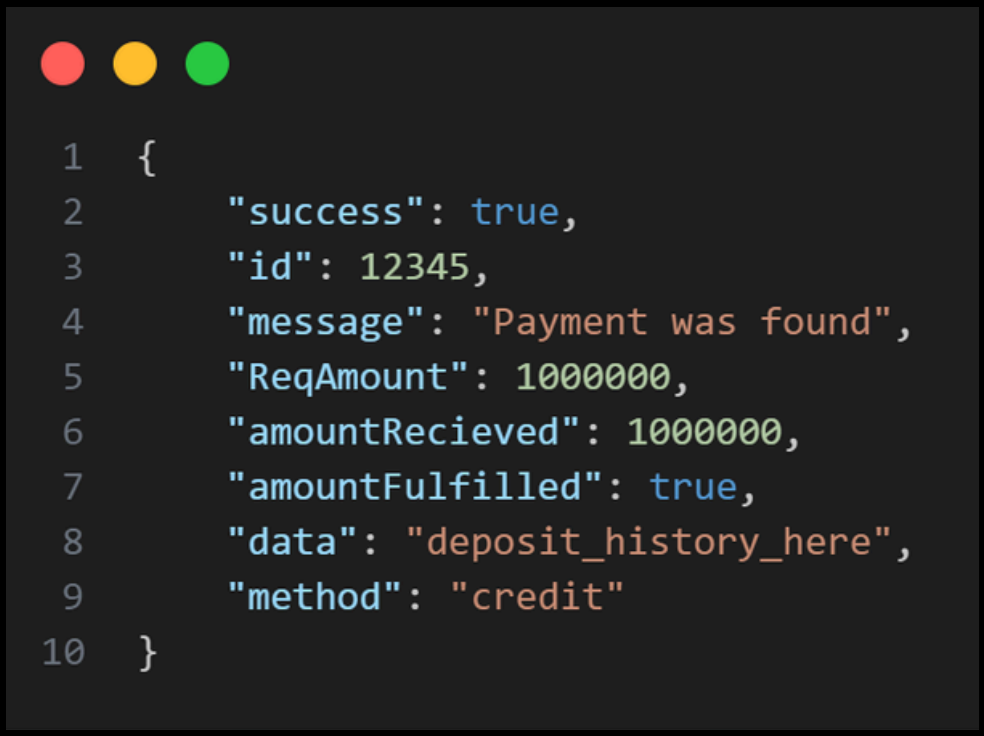
- **Error Responses:**

```
{
    "success": false,
    "message": "Sender and receiver addresses are required"
}
{
    "success": false,
    "message": "Invalid recipient Radiant address"
}
{
    "success": false,
    "message": "No available UTXOs for the given address"
}
{
    "success": false,
    "message": "Insufficient funds for the transaction"
}
{
    "success": false,
    "message": "Internal Server Error",
    "data": "Error message"
}
```

5. Postback System:

- After a successful deposit, the system sends a postback with transaction details to a URL set in the .env file.
- Example Postback Data:

DAPPS



```
{
    "success": true,
    "id": 12345,
    "message": "Payment was found",
    "ReqAmount": 1000000,
    "amountRecieved": 1000000,
    "amountFulfilled": true,
    "data": "deposit_history_here",
    "method": "credit"
}
```

- **Admin Panel Access:**
  - How to Use the Admin Panel:
  - Visit /admin-user-panel for an interface to view key metrics and manage server settings.
  - You can change the ElectrumX server settings, **but remember to restart the app for the changes to take effect.**
  - Important: Admin credentials can be updated in the server.js file. Ensure your credentials are secured.

DAPPS

MuonPay: Community-Driven Funding for Radiant Blockchain

The Radiant Blockchain is 100% community owned and funded. No premine, no ico, no dev fee. All growth, progress and advancements come from the pockets of the community, usually through donation drives or a select few miners. Yet for RXD to reach new heights, steady streams of funding must be found from a larger pool. MuonPay begins to solve this by coming with an encoded soft fee of 0.1% on all deposits, paid by the customer. By everyone pitching in just a tiny portion, not only do we share the load but grow faster together. This fee goes directly to the RXDDapps Development Fund and is solely used for creating more open source Radiant tools and tech like MuonPay. Being true to open source, MuonPay will work even without the fee, if you know how to remove it. But if you truly want Radiant to prosper, you will keep the soft fee. And if you want a truly Radiant future, you will empower your customers to donate even more with the built in deposit donation tool. Radiant belongs to us all. The builders, the miners, the vendors and the buyers. Every shrimp, shark and whale. All of us. We are all the Radiant community and we all must do our part to succeed. Be sure to do yours. Thank you.

rxddapps.com

DAPPS

- **Running the Server:**
  - **Steps to Start the Server:**
  - **Install Node.js v21+.**
  - **Make sure to upload the db.sql file to your sql server**
  - **Run the following command to start the server:**

npm run server

- **All deposits are automatically transaferred to default address (.env)**
- **Contact Information:**
  - **For integration help or any other queries, you can contact RXDDApps directly.**
  - **rxddapps.com**
  - **contact@rxddapps.com**

Built with ❤️ by ThinkDot Solutions. 🚀

https://thinkdotsolutions.com