Here's a comprehensive WebSocket API documentation for the UI team:

```
# WebSocket API Documentation

## Table of Contents
1. [Connection](#connection)
2. [Message Format](#message-format)
3. [Authentication](#authentication)
4. [Request Types](#request-types)
   - [Verification Operations](#verification-operations)
   - [Partner Operations](#partner-operations)
   - [Account Operations](#account-operations)
   - [Deposit Operations](#deposit-operations)
   - [Withdrawal Operations](#withdrawal-operations)
   - [Transfer Operations](#transfer-operations)
   - [Transaction History](#transaction-history)
   - [Statements & Reports](#statements--reports)
   - [Fee Calculation](#fee-calculation)
5. [Server-Sent Events (Push Notifications)](#server-sent-events-push-notifications)
6. [Error Handling](#error-handling)
7. [Examples](#examples)

---

## Connection

### WebSocket Endpoint
```

ws://{{appURL}}/cashier/svc/ws

```
### Connection Headers
```

Authorization: Bearer <your_jwt_token>

~~~
### Connection Lifecycle

**On Connect:**
```json
{
  "type": "connected",
  "data": {
    "user_id": "123",
    "timestamp": 1702000000
  },
  "timestamp": 1702000000
}
~~~

On Disconnect:
Connection is closed. Client should implement reconnection logic with exponential backoff.

Message Format

Request Format

All client messages follow this structure:

```json
{
  "type": "message_type",
  "data": {
    // Request-specific data
  }
}
```

Response Format

Success Response:

```json
{
  "type": "success",
  "data": {
    "message": "Operation successful",
    "data": {
      // Response data
    }
  },
  "timestamp": 1702000000
}
```

Error Response:

```json
{
  "type": "error",
  "data": {
    "message": "Error description"
  },
  "timestamp": 1702000000
}
```

Authentication

Authentication is handled via JWT token in the WebSocket connection headers. No additional authentication is required for individual messages.

Request Types

Verification Operations

1. Request Verification

Used before sensitive operations like withdrawals.

Request:

```
{
  "type": "verification_request",
  "data": {
    "method": "otp_email",  // Options: "totp", "otp_email", "otp_sms", "otp_whatsapp"
    "purpose": "withdrawal"
  }
}
```

Response (OTP):

```
{
  "type": "success",
  "data": {
    "message": "OTP sent to us***@example.com via email",
    "data": {
      "method": "otp_email",
      "channel": "email",
      "recipient": "us***@example.com",
      "purpose": "withdrawal",
      "next_step": "verify_otp",
      "expires_in": 180
    }
  },
  "timestamp": 1702000000
}
```

Response (TOTP - 2FA):

```
{
  "type": "success",
  "data": {
    "message": "2FA enabled.  Please provide your TOTP code",
    "data": {
      "method": "totp",
      "purpose": "withdrawal",
      "next_step": "verify_totp"
    }
  },
  "timestamp": 1702000000
}
```

Error (2FA Not Enabled):

```
{
  "type": "error",
  "data": {
    "message": "2FA is not enabled for your account.  Please use OTP verification instead."
  },
  "timestamp": 1702000000
}
```

2. Verify OTP

Request:

```
{
  "type": "verify_otp",
  "data": {
    "code": "123456",
    "purpose": "withdrawal"
  }
}
```

Success Response:

```
{
  "type": "success",
  "data": {
    "message": "verification successful",
    "data": {
      "verification_token": "a1b2c3d4e5f6789.. .",
      "purpose": "withdrawal",
      "method": "otp_email",
      "expires_in": 300,
      "message": "Use this token for your next withdrawal request"
    }
  },
  "timestamp": 1702000000
}
```

Error Response:

```json
{
  "type": "error",
  "data": {
    "message": "invalid OTP code"
  },
  "timestamp": 1702000000
}
```

3. Verify TOTP (2FA)

Request:

```json
{
  "type": "verify_totp",
  "data": {
    "code": "123456",
    "purpose": "withdrawal"
  }
}
```

Success Response:

```json
{
  "type": "success",
  "data": {
    "message": "verification successful",
    "data": {
      "verification_token": "a1b2c3d4e5f6789...",
      "purpose": "withdrawal",
      "method": "totp",
      "expires_in": 300,
      "message": "Use this token for your next withdrawal request"
    }
  },
  "timestamp": 1702000000
}
```

Partner Operations

1. Get Partners by Service

Request:

```
{
  "type": "get_partners",
  "data": {
    "service": "mpesa"  // Options: mpesa, paypal, bank_transfer, etc.
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "partners retrieved",
    "data": {
      "partners": [
        {
          "id": "PTN-123",
          "name": "Partner Name",
          "country": "KE",
          "service": "mpesa",
          "status": "active"
        }
      ],
      "count": 1
    }
  },
  "timestamp": 1702000000
}
```

Account Operations

1. Get User Accounts

Request:

```json
{
  "type": "get_accounts",
  "data": {}
}
```

Response:

```json
{
  "type": "success",
  "data": {
    "message": "accounts retrieved",
    "data": {
      "accounts": [
        {
          "id": 1,
          "account_number": "ACC-USER-USD-123",
          "currency": "USD",
          "purpose": "ACCOUNT_PURPOSE_WALLET",
          "account_type": "ACCOUNT_TYPE_REAL",
          "is_active": true,
          "is_locked": false,
          "created_at": "2025-01-01T00:00:00Z"
        },
        {
          "id": 2,
          "account_number": "ACC-USER-EUR-124",
          "currency": "EUR",
          "purpose": "ACCOUNT_PURPOSE_WALLET",
          "account_type": "ACCOUNT_TYPE_REAL",
          "is_active": true,
          "is_locked": false,
          "created_at": "2025-01-01T00:00:00Z"
        }
      ],
      "count": 2
    }
  },
  "timestamp": 1702000000
}
```

2. Get Account Balance

Request:

```
{
  "type": "get_account_balance",
  "data": {
    "account_number": "ACC-USER-USD-123"
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "balance retrieved",
    "data": {
      "account_number": "ACC-USER-USD-123",
      "balance": 1000. 50,
      "available_balance": 950.00,
      "pending_debit": 50.50,
      "pending_credit": 0.00,
      "currency": "USD",
      "last_transaction": "2025-12-07T10:30:00Z"
    }
  },
  "timestamp": 1702000000
}
```

3. Get Owner Summary

Get consolidated balance across all accounts.

Request:

```
{
  "type": "get_owner_summary",
  "data": {}
}
```

Response:

```json
{
  "type": "success",
  "data": {
    "message": "owner summary",
    "data": {
      "account_balances": [
        {
          "account_number": "ACC-USER-USD-123",
          "currency": "USD",
          "balance": 1000.50,
          "available_balance": 950.00
        },
        {
          "account_number": "ACC-USER-EUR-124",
          "currency": "EUR",
          "balance": 500.00,
          "available_balance": 500.00
        }
      ],
      "total_balance_usd": 1550.75,
      "total_accounts": 2
    }
  },
  "timestamp": 1702000000
}
```

Deposit Operations

1. Create Deposit Request

Request:

```
{
  "type": "deposit_request",
  "data": {
    "amount": 100.00,
    "currency": "USD",
    "service": "mpesa",
    "partner_id": "PTN-123",  // Optional: will auto-select if not provided
    "payment_method": "mobile_money"  // Optional
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "deposit request created",
    "data": {
      "request_ref": "DEP-REQ-20251207-001",
      "partner_id": "PTN-123",
      "partner_name": "Partner Name",
      "amount": 100.00,
      "currency": "USD",
      "status": "sent_to_partner",
      "expires_at": "2025-12-07T11:00:00Z"
    }
  },
  "timestamp": 1702000000
}
```

2. Get Deposit Status

Request:

```
{
  "type": "get_deposit_status",
  "data": {
    "request_ref": "DEP-REQ-20251207-001"
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "deposit status",
    "data": {
      "request_ref": "DEP-REQ-20251207-001",
      "status": "completed",
      "amount": 100.00,
      "currency": "USD",
      "receipt_code": "RCP-2025-388065207815057408",
      "journal_id": 10,
      "created_at": "2025-12-07T10:30:00Z",
      "completed_at": "2025-12-07T10:35:00Z"
    }
  },
  "timestamp": 1702000000
}
```

3. Cancel Deposit

Request:

```
{
  "type": "cancel_deposit",
  "data": {
    "request_ref": "DEP-REQ-20251207-001"
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "deposit cancelled",
    "data": null
  },
  "timestamp": 1702000000
}
```

Withdrawal Operations

1. Create Withdrawal Request

⚠️ Requires verification token from verification flow

Request:

```
{
  "type": "withdraw_request",
  "data": {
    "amount": 50.00,
    "currency": "USD",
    "destination": "+254712345678",
    "service": "mpesa",  // Optional
    "agent_id": "AGT-123",  // Optional: for agent-assisted withdrawals
    "verification_token": "a1b2c3d4e5f6789..."  // Required: from verification flow
  }
}
```

Response (Success):

```json
{
  "type": "success",
  "data": {
    "message": "withdrawal request created and being processed",
    "data": {
      "request_ref": "WDL-REQ-20251207-001",
      "amount": 50.00,
      "currency": "USD",
      "destination": "+254712345678",
      "status": "processing",
      "withdrawal_type": "direct",  // or "agent_assisted"
      "created_at": 1702000000
    }
  },
  "timestamp": 1702000000
}
```

Response (With Agent):

```json
{
  "type": "success",
  "data": {
    "message": "withdrawal request created and being processed",
    "data": {
      "request_ref": "WDL-REQ-20251207-002",
      "amount": 50.00,
      "currency": "USD",
      "destination": "+254712345678",
      "status": "processing",
      "agent_id": "AGT-123",
      "agent_name": "John's Agent Shop",
      "agent_account": "ACC-AGENT-USD-789",
      "withdrawal_type": "agent_assisted",
      "created_at": 1702000000
    }
  },
  "timestamp": 1702000000
}
```

Error (No Verification Token):

```json
{
  "type": "error",
  "data": {
    "message": "verification_token is required.  Please complete verification first."
  },
  "timestamp": 1702000000
}
```

Error (Invalid Token):

```json
{
  "type": "error",
  "data": {
    "message": "invalid or expired verification token.  Please verify again."
  },
  "timestamp": 1702000000
}
```

Error (Insufficient Balance):

```json
{
  "type": "error",
  "data": {
    "message": "insufficient balance: available 30.00 USD"
  },
  "timestamp": 1702000000
}
```

Transfer Operations

1. Peer-to-Peer Transfer

Request:

```
{
  "type": "transfer",
  "data": {
    "to_user_id": "456",
    "amount": 25.00,
    "currency": "USD",
    "description": "Payment for lunch"
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "transfer completed",
    "data": {
      "receipt_code": "RCP-2025-388065207815057409",
      "journal_id": 11,
      "amount": 25.00,
      "fee": 0.50,
      "agent_commission": 0.00,
      "created_at": "2025-12-07T10:40:00Z"
    }
  },
  "timestamp": 1702000000
}
```

2. Currency Conversion & Transfer

Request:

```
{
  "type": "convert_and_transfer",
  "data": {
    "from_currency": "USD",
    "to_currency": "EUR",
    "amount": 100. 00,
    "description": "Convert USD to EUR"
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "conversion completed",
    "data": {
      "receipt_code": "RCP-2025-388065207815057410",
      "journal_id": 12,
      "source_currency": "USD",
      "dest_currency": "EUR",
      "source_amount": 100.00,
      "converted_amount": 92.50,
      "fx_rate": "0.925",
      "fx_rate_id": 5,
      "fee": 1.00,
      "created_at": "2025-12-07T10:45:00Z"
    }
  },
  "timestamp": 1702000000
}
```

Transaction History

1. Get Transaction History

Request:

```
{
  "type": "get_history",
  "data": {
    "type": "all",  // Options: "deposits", "withdrawals", "all"
    "limit": 20,
    "offset": 0
  }
}
```

Response:

```json
{
  "type": "success",
  "data": {
    "message": "transaction history",
    "data": {
      "deposits": [
        {
          "id": 1,
          "request_ref": "DEP-REQ-20251207-001",
          "amount": 100.00,
          "currency": "USD",
          "status": "completed",
          "receipt_code": "RCP-2025-388065207815057408",
          "created_at": "2025-12-07T10:30:00Z",
          "completed_at": "2025-12-07T10:35:00Z"
        }
      ],
      "withdrawals": [
        {
          "id": 2,
          "request_ref": "WDL-REQ-20251207-001",
          "amount": 50.00,
          "currency": "USD",
          "destination": "+254712345678",
          "status": "completed",
          "receipt_code": "RCP-2025-388065207815057411",
          "created_at": "2025-12-07T11:00:00Z",
          "completed_at": "2025-12-07T11:05:00Z"
        }
      ]
    }
  },
  "timestamp": 1702000000
}
```

2. Get Transaction by Receipt

Request:

```
{
  "type": "get_transaction_by_receipt",
  "data": {
    "receipt_code": "RCP-2025-388065207815057408"
  }
}
```

Response:

```json
{
  "type": "success",
  "data": {
    "message": "transaction details",
    "data": {
      "journal": {
        "id": 10,
        "transaction_type": "TRANSACTION_TYPE_DEPOSIT",
        "description": "Deposit from Partner Name",
        "created_at": "2025-12-07T10:35:00Z"
      },
      "ledgers": [
        {
          "account_number": "ACC-PARTNER-USD-999",
          "amount": 100.00,
          "type": "DR_CR_DEBIT",
          "balance_after": 900.00,
          "description": "Deposit to user"
        },
        {
          "account_number": "ACC-USER-USD-123",
          "amount": 99.00,
          "type": "DR_CR_CREDIT",
          "balance_after": 1099.00,
          "description": "Deposit received"
        }
      ],
      "fees": [
        {
          "type": "FEE_TYPE_PLATFORM",
          "amount": 1.00,
          "currency": "USD"
        }
      ]
    }
  },
  "timestamp": 1702000000
}
```

Statements & Reports

1. Get Account Statement

Request:

```json
{
  "type": "get_account_statement",
  "data": {
    "account_number": "ACC-USER-USD-123",
    "from": "2025-12-01T00:00:00Z",
    "to": "2025-12-07T23:59:59Z"
  }
}
```

Response:

```json
{
  "type": "success",
  "data": {
    "message": "account statement",
    "data": {
      "account_number": "ACC-USER-USD-123",
      "opening_balance": 1000.00,
      "closing_balance": 1099.00,
      "total_debits": 50.00,
      "total_credits": 149.00,
      "period_start": "2025-12-01T00:00:00Z",
      "period_end": "2025-12-07T23:59:59Z",
      "ledgers": [
        {
          "id": 1,
          "amount": 100.00,
          "type": "DR_CR_CREDIT",
          "currency": "USD",
          "balance_after": 1100.00,
          "description": "Deposit received",
          "receipt_code": "RCP-2025-388065207815057408",
          "created_at": "2025-12-07T10:35:00Z"
        },
        {
          "id": 2,
          "amount": 50.00,
          "type": "DR_CR_DEBIT",
          "currency": "USD",
          "balance_after": 1050.00,
          "description": "Withdrawal",
          "receipt_code": "RCP-2025-388065207815057411",
          "created_at": "2025-12-07T11:05:00Z"
        }
      ],
      "transaction_count": 2
    }
  },
  "timestamp": 1702000000
}
```

2. Get Owner Statement

Get statement for all accounts.

Request:

```json
{
  "type": "get_owner_statement",
  "data": {
    "from": "2025-12-01T00:00:00Z",
    "to": "2025-12-07T23:59:59Z"
  }
}
```

Response:

```json
{
  "type": "success",
  "data": {
    "message": "owner statement",
    "data": {
      "statements": [
        {
          "account_number": "ACC-USER-USD-123",
          "opening_balance": 1000.00,
          "closing_balance": 1099.00,
          "total_debits": 50.00,
          "total_credits": 149.00,
          "ledgers": [...]
        },
        {
          "account_number": "ACC-USER-EUR-124",
          "opening_balance": 500. 00,
          "closing_balance": 500.00,
          "total_debits": 0.00,
          "total_credits": 0.00,
          "ledgers": []
        }
      ],
      "count": 2,
      "period_start": "2025-12-01T00:00:00Z",
      "period_end": "2025-12-07T23:59:59Z"
    }
  },
  "timestamp": 1702000000
}
```

3. Get Ledgers

Request:

```
{
  "type": "get_ledgers",
  "data": {
    "account_number": "ACC-USER-USD-123",
    "from": "2025-12-01T00:00:00Z",  // Optional
    "to": "2025-12-07T23:59:59Z",  // Optional
    "limit": 50,
    "offset": 0
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "ledgers retrieved",
    "data": {
      "ledgers": [
        {
          "id": 1,
          "journal_id": 10,
          "amount": 100.00,
          "type": "DR_CR_CREDIT",
          "currency": "USD",
          "balance_after": 1100.00,
          "description": "Deposit received",
          "receipt_code": "RCP-2025-388065207815057408",
          "created_at": "2025-12-07T10:35:00Z"
        }
      ],
      "total": 1,
      "limit": 50,
      "offset": 0
    }
  },
  "timestamp": 1702000000
}
```

Fee Calculation

Calculate Transaction Fee

Request:

```
{
  "type": "calculate_fee",
  "data": {
    "transaction_type": "withdrawal",  // Options: transfer, withdrawal, conversion
    "amount": 100.00,
    "source_currency": "USD",
    "target_currency": "EUR"  // Required for conversion
  }
}
```

Response:

```
{
  "type": "success",
  "data": {
    "message": "fee calculated",
    "data": {
      "fee_type": "FEE_TYPE_PLATFORM",
      "amount": 2.00,
      "currency": "USD",
      "applied_rate": "0.02"
    }
  },
  "timestamp": 1702000000
}
```

Server-Sent Events (Push Notifications)

These are events pushed from the server to connected clients without a request.

1. Deposit Completed

Sent when a deposit is completed by the partner service.

```json
{
  "type": "deposit_completed",
  "data": {
    "transaction_ref": "DEP-REQ-20251207-001",
    "receipt_code": "RCP-2025-388065207815057408",
    "journal_id": 10,
    "amount": 100.00,
    "currency": "USD",
    "user_balance": 1100.00,
    "fee_amount": 1.00,
    "payment_method": "mpesa",
    "completed_at": 1702000000,
    "timestamp": 1702000000
  }
}
```

2. Deposit Failed

Sent when a deposit fails.

```json
{
  "type": "deposit_failed",
  "data": {
    "transaction_ref": "DEP-REQ-20251207-001",
    "amount": 100.00,
    "currency": "USD",
    "error_message": "Payment timeout",
    "failed_at": 1702000000,
    "timestamp": 1702000000
  }
}
```

3. Withdrawal Completed

Sent when a withdrawal is completed.

```json
{
  "type": "withdrawal_completed",
  "data": {
    "request_ref": "WDL-REQ-20251207-001",
    "receipt_code": "RCP-2025-388065207815057411",
    "balance_after": 1050.00
  }
}
```

With Agent:

```json
{
  "type": "withdrawal_completed",
  "data": {
    "request_ref": "WDL-REQ-20251207-002",
    "receipt_code": "RCP-2025-388065207815057412",
    "agent_id": "AGT-123",
    "agent_name": "John's Agent Shop",
    "fee_amount": 2.00,
    "agent_commission": 1.50
  }
}
```

4. Withdrawal Failed

```json
{
  "type": "withdrawal_failed",
  "data": {
    "request_ref": "WDL-REQ-20251207-001",
    "error": "Destination account invalid"
  }
}
```

5. Transfer Completed

Sent when a P2P transfer completes.

```json
{
  "type": "transfer_completed",
  "data": {
    "receipt_code": "RCP-2025-388065207815057409",
    "transaction_id": 11,
    "amount": 25.00,
    "currency": "USD",
    "from_account": "ACC-USER-USD-123",
    "to_account": "ACC-USER-USD-456",
    "fee": 0.50,
    "timestamp": 1702000000
  }
}
```

6. Transaction Completed

Generic transaction completion event.

```json
{
  "type": "transaction_completed",
  "data": {
    "receipt_code": "RCP-2025-388065207815057408",
    "transaction_id": 10,
    "transaction_type": "deposit",
    "amount": 100.00,
    "currency": "USD",
    "balance_after": 1100.00,
    "fee": 1.00,
    "timestamp": 1702000000
  }
}
```

7. Transaction Failed

Generic transaction failure event.

```
{
  "type": "transaction_failed",
  "data": {
    "receipt_code": "RCP-2025-388065207815057408",
    "transaction_id": 10,
    "transaction_type": "deposit",
    "amount": 100.00,
    "currency": "USD",
    "error": "Transaction timeout",
    "timestamp": 1702000000
  }
}
```

Error Handling

Common Error Codes

Error Message	Meaning	Action
`invalid request format`	JSON parsing failed	Check request format
`unauthorized`	Not authorized for this action	Check authentication
`invalid API credentials`	API key/secret invalid	Re-authenticate
`verification_token is required`	Missing verification token	Complete verification flow first
`invalid or expired verification token`	Token invalid/expired	Request new verification
`insufficient balance`	Not enough funds	Add funds or reduce amount
`account not found`	Account doesn't exist	Check account number
`user has no accounts`	No accounts created yet	Create account first
`transaction not found`	Invalid receipt/ref	Check transaction reference

Error Message	Meaning	Action
`amount must be greater than zero`	Invalid amount	Provide valid amount

Examples

Complete Withdrawal Flow

Step 1: Request Verification

```
// Client sends:
{
  "type": "verification_request",
  "data": {
    "method": "otp_email",
    "purpose": "withdrawal"
  }
}

// Server responds:
{
  "type": "success",
  "data": {
    "message": "OTP sent to us***@example.com via email",
    "data": {
      "method": "otp_email",
      "next_step": "verify_otp",
      "expires_in": 180
    }
  }
}
```

Step 2: Verify OTP

```
// Client sends:
{
  "type": "verify_otp",
  "data": {
    "code": "123456",
    "purpose": "withdrawal"
  }
}

// Server responds:
{
  "type": "success",
  "data": {
    "message": "verification successful",
    "data": {
      "verification_token": "abc123xyz.. .",
      "expires_in": 300
    }
  }
}
```

Step 3: Submit Withdrawal

```
// Client sends:
{
  "type": "withdraw_request",
  "data": {
    "amount": 50.00,
    "currency": "USD",
    "destination": "+254712345678",
    "verification_token": "abc123xyz..."
  }
}

// Server responds:
{
  "type": "success",
  "data": {
    "message": "withdrawal request created and being processed",
    "data": {
      "request_ref": "WDL-REQ-20251207-001",
      "status": "processing"
    }
  }
}
```

Step 4: Server Pushes Completion

```
// Server pushes (no request):
{
  "type": "withdrawal_completed",
  "data": {
    "request_ref": "WDL-REQ-20251207-001",
    "receipt_code": "RCP-2025-388065207815057411",
    "balance_after": 1050.00
  }
}
```

Complete Deposit Flow

Step 1: Request Deposit

```
// Client sends:
{
  "type": "deposit_request",
  "data": {
    "amount": 100.00,
    "currency": "USD",
    "service": "mpesa"
  }
}

// Server responds:
{
  "type": "success",
  "data": {
    "message": "deposit request created",
    "data": {
      "request_ref": "DEP-REQ-20251207-001",
      "partner_id": "PTN-123",
      "status": "sent_to_partner",
      "expires_at": "2025-12-07T11:00:00Z"
    }
  }
}
```

Step 2: Server Pushes Completion (When Partner Confirms)

```
// Server pushes (no request):
{
  "type": "deposit_completed",
  "data": {
    "transaction_ref": "DEP-REQ-20251207-001",
    "receipt_code": "RCP-2025-388065207815057408",
    "amount": 100.00,
    "currency": "USD",
    "user_balance": 1100.00,
    "completed_at": 1702000000
  }
}
```

Best Practices

Client Implementation

1. **Reconnection Strategy**
   - Implement exponential backoff for reconnection
   - Start with 1s, then 2s, 4s, 8s, max 30s
   - Reset backoff on successful connection
2. **Message Handling**
   - Always check `type` field first
   - Handle both request/response and push events
   - Store verification tokens securely in memory (not localStorage)
   - Clear verification tokens after use or on expiry
3. **Error Handling**
   - Display user-friendly error messages
   - Log errors for debugging
   - Retry failed operations with user confirmation
4. **State Management**
   - Track connection state (connecting, connected, disconnected)
   - Queue messages while disconnected
   - Update UI based on push events
5. **Security**
   - Never log sensitive data (tokens, passwords)
   - Verify SSL/TLS connection
   - Clear sensitive data on logout

Rate Limits

- Maximum 100 requests per minute per user
- WebSocket connection timeout: 60 seconds of inactivity
- Ping/pong every 54 seconds to keep connection alive

Last Updated: December 7, 2025

API Version: 1.0

This documentation provides:
- ✅ Complete request/response formats for all operations
- ✅ Push notification formats for real-time events
- ✅ Complete verification flow documentation
- ✅ Error handling guide
- ✅ Practical examples
- ✅ Best practices for client implementation
- ✅ Focus on UI team needs (message formats, not backend implementation)