WooCommerce Customer Care MCP Server

A comprehensive **Model Context Protocol (MCP) Server** that provides advanced customer care capabilities for WooCommerce stores. This server enables AI assistants and customer service agents to perform complete order management, customer support, and e-commerce operations through a unified API with **enterprise-level security**.

🔐 **Privacy-First Design**: Built with customer verification to ensure customers can only access their order information, protecting sensitive data and maintaining compliance with privacy regulations.

Latest Updates - v2.0 Security Enhancement

NEW: Customer Verification & Privacy Protection System

- **3 New Security Tools**: verify_customer_for_order, verify_customer_identity, get_verified_order
- **Multi-Factor Verification**: Email + Phone + Name + Address verification
- **Privacy Protection**: Customers can only access their orders
- **Enterprise Security**: GDPR compliance and audit logging
- **Progressive Security**: Different verification levels for different operations

Total: 23 Comprehensive Customer Care Tools (20 original + 3 new security tools

---

Features

Customer Verification & Privacy Protection ⭐ NEW

- Multi-factor customer verification before order access
- Privacy protection - customers can only access their orders
- Secure identity verification with email, phone, name, and address matching
- Progressive verification levels for different operations
- Enterprise-level security compliance (GDPR ready)
- View and search orders with advanced filtering
- Update order status and add notes
- Track order history and customer information
- Process order modifications and status changes

Product Management

- Search products by name, SKU, or description
- Check real-time stock availability
- Get complete product information and pricing
- Filter products by category and status

Customer Management

- Search and manage customer accounts
- View complete customer purchase history
- Create new customer accounts
- Access customer contact and billing information

Customer Service Tools

- Create discount coupons for service recovery
- Process full or partial refunds

- Add internal and customer-visible order notes
- Handle customer complaints and issues

Complete Tool Suite

23 Comprehensive Customer Care Tools:

- **3 Customer Verification Tools** - Secure identity verification and privacy protection
- **7 Order Management Tools** - Complete order lifecycle management
- **3 Product Management Tools** - Inventory and product information
- **4 Customer Management Tools** - Account management and history
- **5 Customer Service Tools** - Service recovery and support operations
- **1 System Tool** - Connectivity testing and diagnostics

Installation

Prerequisites

- Node.js 22+
- npm or yarn
- WooCommerce store with REST API enabled
- WooCommerce API credentials (Consumer Key & Secret)

Getting Started

After purchasing the WooCommerce Customer Care MCP Server, you will receive a source code package. Follow the installation guide below based on your hosting environment.

Local PC Installation

Windows Installation

1. **Extract the source code:**
   - Extract the downloaded ZIP file to your desired location (e.g., C:\woocommerce-mcp-server\)
2. **Install Node.js:**
   - Download and install Node.js 22+ from nodejs.org
   - Verify installation: Open Command Prompt and run node --version
3. **Setup the application:**

   ```
   cd C:\woocommerce-mcp-server
   npm install
   ```

4. **Configure environment variables:**
   - Create a .env file in the root directory:

```
WOOCOMMERCE_URL=https://your-woocommerce-store.com
WOOCOMMERCE_CONSUMER_KEY=ck_your_consumer_key_here
WOOCOMMERCE_CONSUMER_SECRET=cs_your_consumer_secret_here
PORT=3001
```

5. **Build and start:**

```
npm run build
npm start
```

macOS/Linux Installation

1. **Extract the source code:**

```
unzip woocommerce-mcp-server.zip
cd woocommerce-mcp-server
```

2. **Install dependencies:**

```
npm install
```

3. **Configure environment variables:**
   - Create a .env file:

```
WOOCOMMERCE_URL=https://your-woocommerce-store.com
WOOCOMMERCE_CONSUMER_KEY=ck_your_consumer_key_here
WOOCOMMERCE_CONSUMER_SECRET=cs_your_consumer_secret_here
PORT=3001
```

4. **Build and start:**

```
npm run build
npm start
```

The server will start on http://localhost:3001 with the MCP endpoint at /mcp.

VPS Server Installation

Ubuntu/Debian Server

1. **Upload and extract source code:**

   ```
   # Upload the ZIP file to your server via SCP/SFTP
   unzip woocommerce-mcp-server.zip
   cd woocommerce-mcp-server
   ```

2. **Install Node.js:**

   ```
   # Install Node.js 22+
   curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
   sudo apt-get install -y nodejs
   # Verify installation
   node --version
   npm --version
   ```

3. **Install dependencies:**

   ```
   npm install
   ```

4. **Configure environment variables:**

   ```
   nano .env
   ```

   - Add your configuration:

   ```
   WOOCOMMERCE_URL=https://your-woocommerce-store.com
   WOOCOMMERCE_CONSUMER_KEY=ck_your_consumer_key_here
   WOOCOMMERCE_CONSUMER_SECRET=cs_your_consumer_secret_here
   PORT=3001
   NODE_ENV=production
   ```

5. **Build the application:**
   ```
   npm run build
   ```

6. **Install PM2 for process management:**

```
sudo npm install -g pm2
pm2 start dist/index.js --name "woocommerce-mcp"
pm2 startup
pm2 save
```

7. **Configure firewall (optional):**

   ```
   sudo ufw allow 3001
   ```

CentOS/RHEL Server

1. **Install Node.js:**

   ```
   curl -fsSL https://rpm.nodesource.com/setup_22.x | sudo bash -
   sudo yum install -y nodejs
   ```

2. Follow the same steps as Ubuntu for application setup.

cPanel Installation

Shared Hosting with cPanel

1. **Upload source code:**
   - Log in to cPanel
   - Go to File Manager
   - Navigate to public_html or create a subdirectory
   - Upload and extract the ZIP file
2. **Setup Node.js (if supported):**
   - Go to "Software" → "Setup Node.js App"
   - Create a new application:
     - Node.js version: 18+
     - Application mode: Production
     - Application root: /path/to/woocommerce-mcp-server
     - Application URL: your-domain.com/mcp-server (optional)
     - Startup file: dist/index.js
3. **Install dependencies:**
   - Click "Run NPM Install" in the cPanel Node.js interface
   - Or use Terminal if available:

     ```
     cd /home/username/public_html/woocommerce-mcp-server
     npm install
     ```

4. **Configure environment variables:**
   - In cPanel Node.js App settings, add environment variables:
     - WOOCOMMERCE_URL: https://your-store.com
     - WOOCOMMERCE_CONSUMER_KEY: ck_your_key
     - WOOCOMMERCE_CONSUMER_SECRET: cs_your_secret
     - PORT: 3001 (or assigned port)
5. **Build and start:**

   ```
   npm run build
   ```

   - Start the application from the cPanel Node.js interface

WooCommerce API Setup

Enable WooCommerce REST API

1. **Go to WooCommerce Settings:**
    - Navigate to WooCommerce → Settings → Advanced → REST API
2. **Create API Key:**
    - Click "Add Key."
    - Description: MCP Customer Care Server
    - User: Select an administrator user
    - Permissions: Read/Write
    - Click "Generate API Key."
3. **Copy Credentials:**
    - Consumer Key: ck_...
    - Consumer Secret: cs_...
    - Add these to your .env file

Required Permissions

The API user should have permissions for:

- Orders (read/write)
- Products (read)
- Customers (read/write)
- Coupons (read/write)
- Order notes (read/write)
- Refunds (read/write)

Public Access & Production Deployment

Local Development Access

For local testing and development:

```
# Install localtunnel for temporary public access
npm install -g localtunnel
# Create public tunnel
npx localtunnel --port 3001 --subdomain your-store-name
```

Your server will be available at: https://your-store-name.loca.lt/mcp

Production Deployment Options

- **Option 1: VPS with Custom Domain**
  - Deploy on your VPS server (DigitalOcean, AWS EC2, Linode, etc.)
  - Point your domain/subdomain to the server
  - Use a reverse proxy (Nginx/Apache) for SSL and port management
  - Example: https://mcp.yourdomain.com/mcp
- **Option 2: Shared Hosting with cPanel**
  - Upload to shared hosting that supports Node.js
  - Use cPanel's Node.js app manager
  - Access via: https://yourdomain.com:3001/mcp
- **Option 3: Cloud Platforms**
  - **Heroku**: Easy deployment with git-based workflow
  - **Railway**: Modern deployment platform
  - **Vercel**: For serverless deployment
  - **Netlify**: Functions-based deployment
- **Option 4: Docker Container**
  - Containerize the application
  - Deploy on any Docker-compatible platform
  - Use docker-compose for easy management

Integration

VS Code MCP Extension

1. **Install the MCP extension** in VS Code
2. **Configure MCP settings** (.vscode/mcp.json):

```
{
  "servers": {
    "woocommerce-customer-care": {
      "url": "http://localhost:3001/mcp",
      "type": "http"
    }
  },
  "inputs": []
}
```

Claude Desktop Integration

Add to your Claude Desktop MCP configuration:

```
{
  "mcpServers": {
    "woocommerce-customer-care": {
      "command": "node",
      "args": ["/path/to/woocommerce-mcp-server/dist/index.js"],
      "env": {
        "WOOCOMMERCE_URL": "https://your-store.com",
        "WOOCOMMERCE_CONSUMER_KEY": "ck_...",
        "WOOCOMMERCE_CONSUMER_SECRET": "cs_..."
      }
    }
  }
}
```

HTTP API Integration

For custom integrations, use the HTTP MCP protocol:

```
curl -X POST http://localhost:3001/mcp \
  -H "Content-Type: application/json" \
  -d '{
    "jsonrpc": "2.0",
    "method": "tools/call",
    "params": {
      "name": "get_recent_orders",
      "arguments": {"days": 7}
    },
    "id": 1
  }'
```

Available Tools

- **verify_customer_for_order**: Verify customer identity before providing order information.
- **verify_customer_identity**: Verify customer identity for general customer service access.
- **get_verified_order**: Get order details only after successful customer verification.
- **get_orders**: Get orders with advanced filtering options.
- **get_order**: Get complete details of a specific order.
- **search_orders**: Search orders by customer name, email, or order details.
- **get_orders_by_status**: Get all orders with a specific status.
- **get_recent_orders**: Get recent orders from the last X days.
- **update_order_status**: Update order status and add notes.
- **add_order_note**: Add notes to orders (internal or customer-visible).
- **search_products**: Search products with filters.
- **get_product**: Get complete product information.
- **check_product_stock**: Check product availability and stock levels.
- **search_customers**: Search customers by email, name, or phone.
- **get_customer**: Get complete customer information and purchase history.
- **get_customer_orders**: Get complete order history for a customer.
- **create_customer**: Create a new customer account.
- **create_coupon**: Create discount coupons for customer service.
- **process_refund**: Process full or partial refunds.
- **test_connection**: Test WooCommerce API connectivity.

Usage Examples

Customer Service Scenarios

Scenario 1: Secure Customer Order Inquiry

1. Use verify_customer_for_order with order ID and customer details.
2. If verification passes, get order details.
3. Check order status and provide shipping information.
4. Add a customer note about the inquiry.

Scenario 2: Secure Order Access with Combined Tool

1. Use get_verified_order with order ID and verification data.
2. The tool automatically verifies the customer and returns the order if authorized.
3. Provide order status and details securely.

Scenario 3: Process Return and Refund

1. Use get_order to verify order details.
2. Add an order note about the return request.
3. Use process_refund to issue a refund.
4. Update order status to "refunded."
5. Create a service recovery coupon if needed.

Security

Customer Verification & Privacy Protection

- **Multi-Factor Verification**: Requires at least 2 matching customer details (e.g., email/phone + name/address).
- **Order Privacy Protection**: Customers can only access their own order information.
- **Progressive Security**: Different verification levels for different operations.
- **Access Control**: Automatic denial of access if verification fails.

Security Best Practices

- **Always Use Verification**: Use get_verified_order instead of get_order for customer inquiries.
- **Failed Verification**: Never provide order details if verification fails.
- **Logging**: All verification attempts are logged for security auditing.

API Security

- Uses WooCommerce's built-in authentication.
- Requires a valid Consumer Key and Secret.
- All API calls are logged for audit purposes

Production Deployment

Hosting Requirements

- **Minimum:** 1 vCPU, 512MB RAM, 1GB Storage, Node.js 18+ support.
- **Recommended:** 2 vCPU, 2GB RAM, 5GB Storage, SSL Certificate, custom domain/subdomain.

Docker Deployment

Dockerfile

```
FROM node:18-alpine
WORKDIR /app
# Copy package files
COPY package*.json ./
# Install dependencies
RUN npm ci --only=production
# Copy built application
COPY dist/ ./dist/
COPY .env ./
EXPOSE 3001
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD curl -f http://localhost:3001/ || exit 1
CMD ["node", "dist/index.js"]
```

Docker Compose Setup

```
version: '3.8'
services:
  woocommerce-mcp:
    build: .
    ports:
      - "3001:3001"
    environment:
      - NODE_ENV=production
      - WOOCOMMERCE_URL=https://your-store.com
      - WOOCOMMERCE_CONSUMER_KEY=ck_your_key
      - WOOCOMMERCE_CONSUMER_SECRET=cs_your_secret
```

```
  restart: unless-stopped
  volumes:
    - ./logs:/app/logs
```

Troubleshooting

Common Issues

- **Connection Refused**: Check WOOCOMMERCE_URL and ensure the store is accessible.
- **Authentication Failed**: Verify that your Consumer Key and Secret are correct and have Read/Write permissions.
- **Rate Limiting**: WooCommerce API has a default limit of 100 requests/minute.

Support & Updates

Technical Support

- Email support.
- Comprehensive documentation and video tutorials are available.

Updates & Maintenance

- Free updates for bug fixes and security patches.
- New feature updates and version compatibility.

Development Guidelines

- Use npm run dev for development with auto-reload.
- Run tests with npm test.
- Build for production with npm run build.

Made with ❤️ by The Code Studio