ODK MCP System Implementation Guide

Table of Contents

Introduction

The ODK MCP System is a comprehensive implementation of Open Data Kit (ODK) features using Model Context Protocol (MCP). This system provides a robust platform for data collection, management, and analysis, specifically designed for NGOs, think tanks, CSR firms, and other organizations that need to collect and analyze data in various contexts.

This implementation guide provides detailed instructions for setting up, configuring, and deploying the ODK MCP System. It is intended for system administrators, IT professionals, and technical users who are responsible for implementing and maintaining the system.

Purpose and Scope

The purpose of this guide is to provide comprehensive instructions for implementing the ODK MCP System. It covers all aspects of the implementation process, from system requirements and installation to configuration, security setup, deployment, and maintenance.

The guide is structured to support different deployment scenarios, including local desktop deployment, VM deployment, and integration with AI tools. It also provides guidance on security setup, troubleshooting, and maintenance.

Target Audience

This guide is intended for:

- System administrators responsible for installing and maintaining the system
- IT professionals responsible for configuring and securing the system
- Technical users who need to understand the system architecture and components
- Developers who need to extend or customize the system

Prerequisites

Before implementing the ODK MCP System, you should have:

- Basic understanding of ODK concepts and terminology
- Familiarity with command-line interfaces and basic system administration
- Knowledge of Python and web technologies (for customization and extension)
- Access to the necessary hardware and software resources

System Requirements

Hardware Requirements

The ODK MCP System is designed to run on a variety of hardware configurations, from local desktop computers to virtual machines in the cloud. The minimum hardware requirements are:

- **CPU**: 2 cores (4 cores recommended)

- **RAM**: 4 GB (8 GB recommended)
- **Disk Space**: 10 GB (20 GB recommended)
- **Network**: Internet connection for installation and updates

For production deployments with many users or large datasets, consider increasing the resources accordingly.

Software Requirements

The ODK MCP System requires the following software:

- **Operating System**: Ubuntu 20.04 LTS or later (recommended), or any Linux distribution with Python 3.8+
- **Python**: 3.8 or later
- **Database**: SQLite (included) or Baserow (optional)
- **Web Browser**: Chrome, Firefox, or Edge (latest versions)

Dependencies

The ODK MCP System depends on the following Python packages:

- Flask 2.3.3 or later
- Flask-CORS 4.0.0 or later
- Flask-JWT-Extended 4.5.2 or later
- SQLAlchemy 2.0.19 or later
- Pandas 2.0.3 or later
- NumPy 1.24.4 or later
- Streamlit 1.24.0 or later
- Requests 2.31.0 or later
- Matplotlib 3.7.2 or later
- Seaborn 0.12.2 or later
- SciPy 1.10.1 or later
- Statsmodels 0.14.0 or later

All dependencies are automatically installed during the installation process.

Architecture Overview

The ODK MCP System is built on a modular architecture consisting of three main components:

1. **Model Context Protocols (MCPs)**: Server components that handle specific aspects of the system

2. **Agents**: Specialized components that perform specific tasks
3. **User Interface**: Streamlit-based UI for user interaction

Model Context Protocols (MCPs)

The system includes three core MCPs:

1. **Form Management MCP**: Handles form creation, storage, versioning, and retrieval
2. **Data Collection MCP**: Facilitates data collection and manages synchronization of submissions
3. **Data Aggregation & Analytics MCP**: Manages centralized data storage, querying, export, and analytics

Each MCP is implemented as a Flask application with RESTful APIs for interaction with other components.

Agents

The system includes several specialized agents:

1. **Form Management Agent**: Interfaces with the Form Management MCP
2. **Data Collection Agent**: Interfaces with the Data Collection MCP
3. **Data Aggregation Agent**: Interfaces with the Data Aggregation & Analytics MCP
4. **Data Analysis Agents**:
5. **Data Cleaning & Preprocessing Agent**: Handles data cleaning and preprocessing
6. **Descriptive Analytics Agent**: Performs descriptive analytics
7. **Inferential Statistics Agent**: Performs inferential statistics
8. **Data Exploration Agent**: Facilitates data exploration
9. **Report Generation Agent**: Generates reports from analysis results

User Interface

The system includes a Streamlit-based user interface that provides:

- User authentication (sign-in/sign-out)
- Project management
- Form management
- Data collection
- Data analysis
- Reporting

System Interaction

The components interact as follows:

1. Users interact with the system through the Streamlit UI
2. The UI communicates with the agents to perform specific tasks
3. The agents communicate with the MCPs to access and manipulate data
4. The MCPs store and retrieve data from the database (SQLite or Baserow)

Installation

Local Desktop Installation

To install the ODK MCP System on a local desktop:

1. Clone the repository:

```
git clone https://github.com/your-organization/odk-mcp-system.git
cd odk-mcp-system
```

1. Create and activate a virtual environment:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

1. Install the dependencies:

```
pip install -r requirements.txt
```

1. Initialize the database:

```
python setup.py init
```

1. Start the system:

```
python run.py
```

1. Access the UI:

Open a web browser and navigate to `http://localhost:8501`

VM Installation

To install the ODK MCP System on a virtual machine:

1. Provision a VM with Ubuntu 20.04 LTS or later

2. Install the required packages:

```
sudo apt update
sudo apt install -y python3 python3-pip python3-venv git
```

1. Clone the repository:

```
git clone https://github.com/your-organization/odk-mcp-
system.git
cd odk-mcp-system
```

1. Create and activate a virtual environment:

```
python3 -m venv venv
source venv/bin/activate
```

1. Install the dependencies:

```
pip install -r requirements.txt
```

1. Initialize the database:

```
python setup.py init
```

1. Configure the system for VM deployment:

```
python setup.py configure --deployment=vm
```

1. Start the system:

```
python run.py
```

1. Access the UI:

Open a web browser and navigate to `http://<vm-ip>:8501`

Docker Installation

To install the ODK MCP System using Docker:

1. Install Docker and Docker Compose:

```
sudo apt update
sudo apt install -y docker.io docker-compose
```

1. Clone the repository:

```
git clone https://github.com/your-organization/odk-mcp-system.git
cd odk-mcp-system
```

1. Build and start the containers:

```
docker-compose up -d
```

1. Access the UI:

Open a web browser and navigate to `http://localhost:8501`

Configuration

MCP Configuration

Each MCP can be configured using environment variables or configuration files. The default configuration files are located in the `config` directory.

Form Management MCP

The Form Management MCP can be configured using the following environment variables:

- `FORM_MANAGEMENT_PORT`: Port for the Form Management MCP (default: 5001)
- `FORM_MANAGEMENT_DB_URI`: Database URI for the Form Management MCP (default: `sqlite:///form_management.db`)
- `FORM_MANAGEMENT_JWT_SECRET`: Secret key for JWT authentication (default: randomly generated)

Data Collection MCP

The Data Collection MCP can be configured using the following environment variables:

- `DATA_COLLECTION_PORT`: Port for the Data Collection MCP (default: 5002)
- `DATA_COLLECTION_DB_URI`: Database URI for the Data Collection MCP (default: `sqlite:///data_collection.db`)
- `DATA_COLLECTION_JWT_SECRET`: Secret key for JWT authentication (default: randomly generated)

Data Aggregation & Analytics MCP

The Data Aggregation & Analytics MCP can be configured using the following environment variables:

- `DATA_AGGREGATION_PORT`: Port for the Data Aggregation & Analytics MCP (default: 5003)
- `DATA_AGGREGATION_DB_URI`: Database URI for the Data Aggregation & Analytics MCP (default: `sqlite:///data_aggregation.db`)
- `DATA_AGGREGATION_JWT_SECRET`: Secret key for JWT authentication (default: randomly generated)
- `DATA_AGGREGATION_BASEROW_URL`: URL for Baserow integration (optional)
- `DATA_AGGREGATION_BASEROW_TOKEN`: API token for Baserow integration (optional)

Agent Configuration

The agents can be configured using environment variables or configuration files. The default configuration files are located in the `config` directory.

Data Analysis Agents

The Data Analysis Agents can be configured using the following environment variables:

- `DATA_ANALYSIS_TEMP_DIR`: Directory for temporary files (default: `temp`)
- `DATA_ANALYSIS_MAX_ROWS`: Maximum number of rows to process (default: 10000)
- `DATA_ANALYSIS_TIMEOUT`: Timeout for analysis operations in seconds (default: 300)

UI Configuration

The Streamlit UI can be configured using environment variables or the `.streamlit/config.toml` file.

- `STREAMLIT_SERVER_PORT`: Port for the Streamlit server (default: 8501)
- `STREAMLIT_SERVER_HEADLESS`: Run in headless mode (default: false)
- `STREAMLIT_BROWSER_GATHER_USAGE_STATS`: Gather usage statistics (default: false)

Baserow Integration

To integrate with Baserow:

1. Install and set up Baserow (see Baserow documentation)
2. Create a Baserow API token:
3. Log in to Baserow
4. Go to Account > API tokens
5. Create a new token with appropriate permissions
6. Configure the Data Aggregation & Analytics MCP to use Baserow:

```
export DATA_AGGREGATION_BASEROW_URL=https://your-baserow-
instance.com/api
export DATA_AGGREGATION_BASEROW_TOKEN=your-api-token
```

1. Restart the Data Aggregation & Analytics MCP:

```
python run.py restart data_aggregation
```

Security Setup

Authentication

The ODK MCP System uses JWT (JSON Web Token) for authentication. The authentication flow is as follows:

1. User provides credentials (username and password)
2. System validates credentials and issues a JWT

3. JWT is included in subsequent requests for authentication

To configure authentication:

1. Set a strong JWT secret key:

```
export FORM_MANAGEMENT_JWT_SECRET=your-secret-key
export DATA_COLLECTION_JWT_SECRET=your-secret-key
export DATA_AGGREGATION_JWT_SECRET=your-secret-key
```

1. Configure token expiration:

```
export JWT_ACCESS_TOKEN_EXPIRES=3600  # 1 hour
```

1. Configure password hashing:

```
export PASSWORD_HASH_ROUNDS=12  # Higher is more secure but
slower
```

Authorization

The ODK MCP System uses role-based access control (RBAC) for authorization. The available roles are:

- `ADMIN`: Full access to all features
- `PROJECT_MANAGER`: Can manage projects and forms
- `DATA_COLLECTOR`: Can submit data
- `ANALYST`: Can analyze data
- `VIEWER`: Can view data

To configure authorization:

1. Set default roles for new users:

```
export DEFAULT_USER_ROLE=VIEWER
```

1. Configure project-level roles:

```
export PROJECT_CREATOR_ROLE=PROJECT_MANAGER
```

Data Security

To secure data at rest:

1. Configure database encryption:

```
export DB_ENCRYPTION_KEY=your-encryption-key
```

1. Configure secure file storage:

```
export SECURE_FILE_STORAGE=true
export FILE_ENCRYPTION_KEY=your-encryption-key
```

To secure data in transit:

1. Configure HTTPS:

```
export USE_HTTPS=true
export SSL_CERT_PATH=/path/to/cert.pem
export SSL_KEY_PATH=/path/to/key.pem
```

Deployment

Local Deployment

For local deployment, follow the Local Desktop Installation instructions.

VM Deployment

For VM deployment, follow the VM Installation instructions.

Additionally, consider setting up a reverse proxy (e.g., Nginx) to handle HTTPS and load balancing:

1. Install Nginx:

```
sudo apt update
sudo apt install -y nginx
```

1. Configure Nginx:

```
sudo nano /etc/nginx/sites-available/odk-mcp-system
```

Add the following configuration:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:8501;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /api/form-management/ {
        proxy_pass http://localhost:5001/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /api/data-collection/ {
        proxy_pass http://localhost:5002/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    location /api/data-aggregation/ {
        proxy_pass http://localhost:5003/api/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

1. Enable the site:

```
sudo ln -s /etc/nginx/sites-available/odk-mcp-system /etc/nginx/
sites-enabled/
sudo systemctl restart nginx
```

1. Set up HTTPS with Let's Encrypt:

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d your-domain.com
```

AI Tool Integration

To integrate the ODK MCP System with AI tools like Claude for Desktop or ChatGPT:

1. Configure the system for API access:

```
export ENABLE_API_ACCESS=true
export API_ACCESS_TOKEN=your-api-token
```

1. Create API documentation for the AI tool:

```
python tools/generate_api_docs.py --format=openapi --
output=api_docs.json
```

1. Register the API with the AI tool platform

2. Create tool definitions for the AI tool:

```
python tools/generate_tool_definitions.py --ai-platform=claude
--output=claude_tools.json
```

1. Upload the tool definitions to the AI tool platform

Troubleshooting

Common Issues

Installation Issues

Issue: Dependencies fail to install

Solution: - Check Python version (should be 3.8+) - Update pip: `pip install --upgrade pip` - Install development packages: `sudo apt install -y python3-dev build-essential`

Database Issues

Issue: Database connection errors

Solution: - Check database URI configuration - Ensure database file is writable - For Baserow, check API token and URL

Authentication Issues

Issue: JWT authentication fails

Solution: - Check JWT secret key configuration - Ensure clocks are synchronized - Check token expiration settings

Performance Issues

Issue: System is slow with large datasets

Solution: - Increase hardware resources - Configure data analysis limits - Use database indexing

Logs

Log files are located in the `logs` directory:

- `form_management.log`: Form Management MCP logs
- `data_collection.log`: Data Collection MCP logs
- `data_aggregation.log`: Data Aggregation & Analytics MCP logs
- `ui.log`: Streamlit UI logs

To change log levels:

```
export LOG_LEVEL=DEBUG  # or INFO, WARNING, ERROR
```

Diagnostics

To run system diagnostics:

```
python tools/diagnostics.py
```

This will check: - System requirements - Component connectivity - Database access - Authentication configuration - File permissions

Maintenance

Backup and Restore

To backup the system:

```
python tools/backup.py --output=backup.zip
```

To restore from a backup:

```
python tools/restore.py --input=backup.zip
```

Database Maintenance

To optimize the database:

```
python tools/optimize_db.py
```

To clean up old data:

```
python tools/cleanup.py --older-than=30d
```

Monitoring

To monitor system health:

```
python tools/monitor.py
```

This will show: - CPU and memory usage - Database size and performance - API request rates - Error rates

Upgrading

To upgrade the ODK MCP System:

1. Backup the system:

```
python tools/backup.py --output=backup.zip
```

1. Update the repository:

```
git pull
```

1. Update dependencies:

```
pip install -r requirements.txt
```

1. Run database migrations:

```
python setup.py migrate
```

1. Restart the system:

```
python run.py restart
```

References

1. Open Data Kit Documentation
2. Model Context Protocol Specification
3. Flask Documentation
4. Streamlit Documentation
5. Baserow Documentation
6. SQLite Documentation
7. JWT Authentication
8. RBAC Best Practices
9. Data Security Guidelines
10. API Integration Best Practices