ODK MCP System API Reference

Table of Contents

Introduction

The ODK MCP System API provides programmatic access to the system's functionality. This reference documents the available endpoints, request and response formats, and authentication mechanisms.

Base URLs

The API is divided into three main components, each with its own base URL:

- **Form Management API**: `http://<host>:<port>/api/v1/form-management`
- **Data Collection API**: `http://<host>:<port>/api/v1/data-collection`
- **Data Aggregation API**: `http://<host>:<port>/api/v1/data-aggregation`

Request Format

API requests should be formatted as follows:

- **GET** requests: Parameters should be included in the query string
- **POST**, **PUT**, **PATCH** requests: Parameters should be included in the request body as JSON
- **DELETE** requests: Parameters should be included in the query string

Response Format

All API responses are in JSON format and include the following fields:

- **success**: Boolean indicating whether the request was successful
- **data**: The response data (if the request was successful)
- **error**: Error information (if the request failed)
- **meta**: Metadata about the response (pagination, etc.)

Example successful response:

```json
{
  "success": true,
  "data": {
    "id": 123,
    "name": "Example"
  },
  "meta": {
    "timestamp": "2023-06-12T15:30:45Z"
  }
}
```

Example error response:

```json
{
  "success": false,
  "error": {
    "code": "INVALID_INPUT",
    "message": "Invalid input parameters",
    "details": {
      "field": "name",
      "issue": "Name is required"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T15:30:45Z"
  }
}
```

Authentication

The API uses JWT (JSON Web Token) for authentication. To authenticate:

1. Obtain a token by calling the authentication endpoint
2. Include the token in the `Authorization` header of subsequent requests

Obtaining a Token

Endpoint: `POST /api/v1/data-aggregation/auth/login`

Request Body:

```json
{
  "username": "your-username",
  "password": "your-password"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "expires_at": "2023-06-12T16:30:45Z",
    "user": {
      "id": 123,
      "username": "your-username",
      "role": "ADMIN"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T15:30:45Z"
  }
}
```

Using the Token

Include the token in the `Authorization` header of your requests:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Token Expiration

Tokens expire after a certain period (typically 1 hour). When a token expires, you need to obtain a new one.

API Keys

For server-to-server communication, you can use API keys instead of JWT tokens. To use an API key:

1. Generate an API key in the system settings
2. Include the API key in the `X-API-Key` header of your requests

```
X-API-Key: your-api-key
```

Error Handling

The API uses standard HTTP status codes to indicate the success or failure of requests:

- **2xx**: Success
- **200 OK**: The request was successful
- **201 Created**: A resource was successfully created
- **204 No Content**: The request was successful but there is no content to return
- **4xx**: Client Error
- **400 Bad Request**: The request was invalid
- **401 Unauthorized**: Authentication is required
- **403 Forbidden**: The authenticated user does not have permission
- **404 Not Found**: The requested resource was not found
- **409 Conflict**: The request conflicts with the current state
- **422 Unprocessable Entity**: The request was well-formed but contains semantic errors
- **5xx**: Server Error
- **500 Internal Server Error**: An error occurred on the server
- **503 Service Unavailable**: The service is temporarily unavailable

Error Codes

In addition to HTTP status codes, the API uses the following error codes:

- **AUTHENTICATION_FAILED**: Authentication failed
- **AUTHORIZATION_FAILED**: Authorization failed
- **INVALID_INPUT**: Invalid input parameters
- **RESOURCE_NOT_FOUND**: The requested resource was not found
- **RESOURCE_ALREADY_EXISTS**: The resource already exists
- **VALIDATION_FAILED**: Validation failed
- **INTERNAL_ERROR**: An internal error occurred

Error Details

The `error.details` field provides additional information about the error. The structure of this field depends on the error type.

Form Management API

The Form Management API allows you to manage forms in the system.

List Forms

Endpoint: `GET /api/v1/form-management/forms`

Query Parameters:

- **project_id**: (Optional) Filter forms by project ID
- **page**: (Optional) Page number for pagination (default: 1)
- **per_page**: (Optional) Number of items per page (default: 10)
- **sort_by**: (Optional) Field to sort by (default: "created_at")
- **sort_order**: (Optional) Sort order ("asc" or "desc", default: "desc")

Response:

```
{
  "success": true,
  "data": {
    "forms": [
      {
        "id": 123,
        "name": "Example Form",
        "project_id": 456,
        "version": "1.0",
        "created_at": "2023-06-12T15:30:45Z",
        "created_by": "admin",
        "updated_at": "2023-06-12T15:30:45Z"
      },
      ...
    ]
  },
  "meta": {
    "page": 1,
    "per_page": 10,
    "total_pages": 5,
    "total_items": 42
  }
}
```

Get Form

Endpoint: `GET /api/v1/form-management/forms/{form_id}`

Path Parameters:

- **form_id**: ID of the form to retrieve

Response:

```
  {
    "success": true,
    "data": {
      "form": {
        "id": 123,
        "name": "Example Form",
        "project_id": 456,
        "version": "1.0",
        "created_at": "2023-06-12T15:30:45Z",
        "created_by": "admin",
        "updated_at": "2023-06-12T15:30:45Z",
        "xml_content": "<form>...</form>",
        "json_schema": "{...}"
      }
    },
    "meta": {
      "timestamp": "2023-06-12T15:30:45Z"
    }
  }
```

Create Form

Endpoint: `POST /api/v1/form-management/forms`

Request Body:

```
{
  "name": "New Form",
  "project_id": 456,
  "version": "1.0",
  "xml_content": "<form>...</form>",
  "json_schema": "{...}"
}
```

Response:

```
{
  "success": true,
  "data": {
    "form_id": 123,
    "name": "New Form",
    "project_id": 456,
    "version": "1.0",
    "created_at": "2023-06-12T15:30:45Z",
    "created_by": "admin"
  },
  "meta": {
    "timestamp": "2023-06-12T15:30:45Z"
  }
}
```

Update Form

Endpoint: `PUT /api/v1/form-management/forms/{form_id}`

Path Parameters:

- **form_id**: ID of the form to update

Request Body:

```
{
  "name": "Updated Form",
  "version": "1.1",
  "xml_content": "<form>...</form>",
  "json_schema": "{...}"
}
```

Response:

```
{
  "success": true,
  "data": {
    "form": {
      "id": 123,
      "name": "Updated Form",
      "project_id": 456,
      "version": "1.1",
      "created_at": "2023-06-12T15:30:45Z",
      "created_by": "admin",
      "updated_at": "2023-06-12T15:35:12Z"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T15:35:12Z"
  }
}
```

Delete Form

Endpoint: `DELETE /api/v1/form-management/forms/{form_id}`

Path Parameters:

- **form_id**: ID of the form to delete

Response:

```
{
  "success": true,
  "data": {
    "message": "Form deleted successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T15:40:22Z"
  }
}
```

Upload XLSForm

Endpoint: `POST /api/v1/form-management/forms/upload`

Request Body:

Multipart form data with the following fields: - **name**: Name of the form - **project_id**: ID of the project - **xlsform**: XLSForm file

Response:

```
{
  "success": true,
  "data": {
    "form_id": 123,
    "name": "New Form",
    "project_id": 456,
    "version": "1.0",
    "created_at": "2023-06-12T15:30:45Z",
    "created_by": "admin"
  },
  "meta": {
    "timestamp": "2023-06-12T15:30:45Z"
  }
}
```

Data Collection API

The Data Collection API allows you to manage submissions in the system.

List Submissions

Endpoint: `GET /api/v1/data-collection/submissions`

Query Parameters:

- **form_id**: (Optional) Filter submissions by form ID
- **project_id**: (Optional) Filter submissions by project ID
- **submitted_by**: (Optional) Filter submissions by submitter
- **start_date**: (Optional) Filter submissions by start date
- **end_date**: (Optional) Filter submissions by end date
- **page**: (Optional) Page number for pagination (default: 1)
- **per_page**: (Optional) Number of items per page (default: 10)
- **sort_by**: (Optional) Field to sort by (default: "submitted_at")
- **sort_order**: (Optional) Sort order ("asc" or "desc", default: "desc")

Response:

```
{
  "success": true,
  "data": {
    "submissions": [
      {
        "id": 789,
        "form_id": 123,
        "project_id": 456,
        "submitted_by": "user",
        "submitted_at": "2023-06-12T15:45:30Z",
        "status": "COMPLETE"
      },
      ...
    ]
  },
  "meta": {
    "page": 1,
    "per_page": 10,
    "total_pages": 3,
    "total_items": 27
  }
}
```

Get Submission

Endpoint: `GET /api/v1/data-collection/submissions/{submission_id}`

Path Parameters:

- **submission_id**: ID of the submission to retrieve

Response:

```
  {
    "success": true,
    "data": {
      "submission": {
        "id": 789,
        "form_id": 123,
        "project_id": 456,
        "submitted_by": "user",
        "submitted_at": "2023-06-12T15:45:30Z",
        "status": "COMPLETE",
        "data": {
          "field1": "value1",
          "field2": "value2",
          ...
        }
      }
    },
    "meta": {
      "timestamp": "2023-06-12T15:50:12Z"
    }
  }
```

Create Submission

Endpoint: `POST /api/v1/data-collection/submissions`

Request Body:

```
{
  "form_id": 123,
  "project_id": 456,
  "data": {
    "field1": "value1",
    "field2": "value2",
    ...
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "submission_id": 789,
    "form_id": 123,
    "project_id": 456,
    "submitted_by": "user",
    "submitted_at": "2023-06-12T15:45:30Z",
    "status": "COMPLETE"
  },
  "meta": {
    "timestamp": "2023-06-12T15:45:30Z"
  }
}
```

Update Submission

Endpoint: `PUT /api/v1/data-collection/submissions/{submission_id}`

Path Parameters:

- **submission_id**: ID of the submission to update

Request Body:

```
{
  "data": {
    "field1": "updated_value1",
    "field2": "updated_value2",
    ...
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "submission": {
      "id": 789,
      "form_id": 123,
      "project_id": 456,
      "submitted_by": "user",
      "submitted_at": "2023-06-12T15:45:30Z",
      "updated_at": "2023-06-12T15:55:22Z",
      "status": "COMPLETE"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T15:55:22Z"
  }
}
```

Delete Submission

Endpoint: `DELETE /api/v1/data-collection/submissions/{submission_id}`

Path Parameters:

- **submission_id**: ID of the submission to delete

Response:

```
{
  "success": true,
  "data": {
    "message": "Submission deleted successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T16:00:15Z"
  }
}
```

Bulk Submission

Endpoint: `POST /api/v1/data-collection/submissions/bulk`

Request Body:

```json
{
  "submissions": [
    {
      "form_id": 123,
      "project_id": 456,
      "data": {
        "field1": "value1",
        "field2": "value2",
        ...
      }
    },
    ...
  ]
}
```

Response:

```json
{
  "success": true,
  "data": {
    "submission_ids": [789, 790, 791],
    "failed_submissions": []
  },
  "meta": {
    "timestamp": "2023-06-12T16:05:45Z"
  }
}
```

Sync Submissions

Endpoint: `POST /api/v1/data-collection/submissions/sync`

Request Body:

```json
{
  "project_id": 456,
  "form_id": 123,
  "last_sync": "2023-06-12T15:00:00Z"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "submissions": [
      {
        "id": 789,
        "form_id": 123,
        "project_id": 456,
        "submitted_by": "user",
        "submitted_at": "2023-06-12T15:45:30Z",
        "status": "COMPLETE",
        "data": {
          "field1": "value1",
          "field2": "value2",
          ...
        }
      },
      ...
    ],
    "last_sync": "2023-06-12T16:10:22Z"
  },
  "meta": {
    "timestamp": "2023-06-12T16:10:22Z"
  }
}
```

Data Aggregation API

The Data Aggregation API allows you to manage users, projects, and data in the system.

User Management

Register User

Endpoint: `POST /api/v1/data-aggregation/auth/register`

Request Body:

```json
{
  "username": "new_user",
  "email": "user@example.com",
  "password": "secure_password",
  "role": "USER"
}
```

Response:

```
{
  "success": true,
  "data": {
    "user_id": 123,
    "username": "new_user",
    "email": "user@example.com",
    "role": "USER",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  },
  "meta": {
    "timestamp": "2023-06-12T16:15:30Z"
  }
}
```

Login

Endpoint: `POST /api/v1/data-aggregation/auth/login`

Request Body:

```
{
  "username": "existing_user",
  "password": "secure_password"
}
```

Response:

```
{
  "success": true,
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "expires_at": "2023-06-12T17:15:30Z",
    "user": {
      "id": 123,
      "username": "existing_user",
      "email": "user@example.com",
      "role": "USER"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T16:15:30Z"
  }
}
```

Get User

Endpoint: `GET /api/v1/data-aggregation/users/{user_id}`

Path Parameters:

- **user_id**: ID of the user to retrieve

Response:

```
{
  "success": true,
  "data": {
    "user": {
      "id": 123,
      "username": "existing_user",
      "email": "user@example.com",
      "role": "USER",
      "created_at": "2023-06-12T10:00:00Z",
      "last_login": "2023-06-12T16:15:30Z"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T16:20:15Z"
  }
}
```

Update User

Endpoint: `PUT /api/v1/data-aggregation/users/{user_id}`

Path Parameters:

- **user_id**: ID of the user to update

Request Body:

```
{
  "email": "updated_email@example.com",
  "role": "PROJECT_MANAGER"
}
```

Response:

```
{
  "success": true,
  "data": {
    "user": {
      "id": 123,
      "username": "existing_user",
      "email": "updated_email@example.com",
      "role": "PROJECT_MANAGER",
      "updated_at": "2023-06-12T16:25:45Z"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T16:25:45Z"
  }
}
```

Delete User

Endpoint: `DELETE /api/v1/data-aggregation/users/{user_id}`

Path Parameters:

- **user_id**: ID of the user to delete

Response:

```
{
  "success": true,
  "data": {
    "message": "User deleted successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T16:30:22Z"
  }
}
```

Project Management

List Projects

Endpoint: `GET /api/v1/data-aggregation/projects`

Query Parameters:

- **page**: (Optional) Page number for pagination (default: 1)
- **per_page**: (Optional) Number of items per page (default: 10)
- **sort_by**: (Optional) Field to sort by (default: "created_at")
- **sort_order**: (Optional) Sort order ("asc" or "desc", default: "desc")

Response:

```json
{
  "success": true,
  "data": {
    "projects": [
      {
        "id": 456,
        "name": "Example Project",
        "description": "A project for example purposes",
        "created_at": "2023-06-12T14:00:00Z",
        "created_by": 123,
        "updated_at": "2023-06-12T14:00:00Z"
      },
      ...
    ]
  },
  "meta": {
    "page": 1,
    "per_page": 10,
    "total_pages": 2,
    "total_items": 15
  }
}
```

Get Project

Endpoint: `GET /api/v1/data-aggregation/projects/{project_id}`

Path Parameters:

- **project_id**: ID of the project to retrieve

Response:

```json
{
  "success": true,
  "data": {
    "project": {
      "id": 456,
      "name": "Example Project",
      "description": "A project for example purposes",
      "created_at": "2023-06-12T14:00:00Z",
      "created_by": 123,
      "updated_at": "2023-06-12T14:00:00Z",
      "form_count": 5,
      "submission_count": 27
    }
  },
  "meta": {
    "timestamp": "2023-06-12T16:35:12Z"
  }
}
```

Create Project

Endpoint: `POST /api/v1/data-aggregation/projects`

Request Body:

```json
{
  "name": "New Project",
  "description": "A new project for testing"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "project_id": 457,
    "name": "New Project",
    "description": "A new project for testing",
    "created_at": "2023-06-12T16:40:30Z",
    "created_by": 123
  },
  "meta": {
    "timestamp": "2023-06-12T16:40:30Z"
  }
}
```

Update Project

Endpoint: `PUT /api/v1/data-aggregation/projects/{project_id}`

Path Parameters:

- **project_id**: ID of the project to update

Request Body:

```json
{
  "name": "Updated Project",
  "description": "An updated project description"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "project": {
      "id": 456,
      "name": "Updated Project",
      "description": "An updated project description",
      "created_at": "2023-06-12T14:00:00Z",
      "created_by": 123,
      "updated_at": "2023-06-12T16:45:22Z"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T16:45:22Z"
  }
}
```

Delete Project

Endpoint: `DELETE /api/v1/data-aggregation/projects/{project_id}`

Path Parameters:

- **project_id**: ID of the project to delete

Response:

```json
{
  "success": true,
  "data": {
    "message": "Project deleted successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T16:50:15Z"
  }
}
```

Add User to Project

Endpoint: `POST /api/v1/data-aggregation/projects/{project_id}/users`

Path Parameters:

- **project_id**: ID of the project

Request Body:

```
{
  "user_id": 124,
  "role": "VIEWER"
}
```

Response:

```
{
  "success": true,
  "data": {
    "project_id": 456,
    "user_id": 124,
    "role": "VIEWER",
    "added_at": "2023-06-12T16:55:30Z"
  },
  "meta": {
    "timestamp": "2023-06-12T16:55:30Z"
  }
}
```

Remove User from Project

Endpoint: `DELETE /api/v1/data-aggregation/projects/{project_id}/users/{user_id}`

Path Parameters:

- **project_id**: ID of the project
- **user_id**: ID of the user to remove

Response:

```
{
  "success": true,
  "data": {
    "message": "User removed from project successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T17:00:45Z"
  }
}
```

Data Management

Query Data

Endpoint: `POST /api/v1/data-aggregation/data/query`

Request Body:

```
{
  "project_id": 456,
  "form_id": 123,
  "filters": [
    {
      "field": "field1",
      "operator": "eq",
      "value": "value1"
    }
  ],
  "sort": [
    {
      "field": "submitted_at",
      "order": "desc"
    }
  ],
  "page": 1,
  "per_page": 10
}
```

Response:

```
{
  "success": true,
  "data": {
    "results": [
      {
        "id": 789,
        "form_id": 123,
        "project_id": 456,
        "submitted_by": "user",
        "submitted_at": "2023-06-12T15:45:30Z",
        "data": {
          "field1": "value1",
          "field2": "value2",
          ...
        }
      },
      ...
    ],
    "total": 5
  },
  "meta": {
    "page": 1,
    "per_page": 10,
    "total_pages": 1
  }
}
```

Export Data

Endpoint: `POST /api/v1/data-aggregation/data/export`

Request Body:

```json
{
  "project_id": 456,
  "form_id": 123,
  "format": "csv",
  "filters": [
    {
      "field": "field1",
      "operator": "eq",
      "value": "value1"
    }
  ],
  "fields": ["field1", "field2"]
}
```

Response:

```json
{
  "success": true,
  "data": {
    "export_id": "export_123",
    "format": "csv",
    "url": "https://example.com/exports/export_123.csv",
    "expires_at": "2023-06-13T17:05:30Z"
  },
  "meta": {
    "timestamp": "2023-06-12T17:05:30Z"
  }
}
```

Analysis API

The Analysis API allows you to perform data analysis on the collected data.

Descriptive Analytics

Endpoint: `POST /api/v1/data-aggregation/analysis/descriptive`

Request Body:

```json
{
  "project_id": 456,
  "form_id": 123,
  "variables": ["field1", "field2"],
  "filters": [
    {
      "field": "field3",
      "operator": "gt",
      "value": 10
    }
  ],
  "options": {
    "include_summary": true,
    "include_frequency": true,
    "include_visualizations": true
  }
}
```

Response:

```json
{
  "success": true,
  "data": {
    "analysis_id": "analysis_123",
    "summary_statistics": {
      "field1": {
        "count": 5,
        "mean": 15.2,
        "median": 14,
        "min": 10,
        "max": 22,
        "std": 4.6
      },
      "field2": {
        "count": 5,
        "categories": ["A", "B", "C"],
        "mode": "B"
      }
    },
    "frequency_tables": {
      "field2": {
        "A": 1,
        "B": 3,
        "C": 1
      }
    },
    "visualizations": [
      {
        "type": "histogram",
        "variable": "field1",
        "url": "https://example.com/visualizations/
histogram_field1.png"
      },
      {
        "type": "bar_chart",
        "variable": "field2",
        "url": "https://example.com/visualizations/
bar_chart_field2.png"
      }
    ]
  },
  "meta": {
    "timestamp": "2023-06-12T17:10:45Z"
  }
}
```

Inferential Statistics

Endpoint: `POST /api/v1/data-aggregation/analysis/inferential`

Request Body:

```
{
  "project_id": 456,
  "form_id": 123,
  "analysis_type": "t_test",
  "variables": ["field1"],
  "group_variable": "field2",
  "groups": ["A", "B"],
  "hypothesis": "two_sided",
  "alpha": 0.05,
  "options": {
    "include_visualizations": true
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "analysis_id": "analysis_124",
    "analysis_type": "t_test",
    "result": {
      "statistic": 2.45,
      "p_value": 0.03,
      "significant": true,
      "confidence_interval": [0.5, 9.7],
      "effect_size": 1.2
    },
    "group_statistics": {
      "A": {
        "count": 3,
        "mean": 18.3,
        "std": 3.2
      },
      "B": {
        "count": 2,
        "mean": 10.5,
        "std": 0.7
      }
    },
    "visualizations": [
      {
        "type": "box_plot",
        "variables": ["field1"],
        "group_variable": "field2",
        "url": "https://example.com/visualizations/
box_plot_field1_by_field2.png"
      }
    ]
  },
  "meta": {
    "timestamp": "2023-06-12T17:15:30Z"
  }
}
```

Data Exploration

Endpoint: `POST /api/v1/data-aggregation/analysis/explore`

Request Body:

```
{
  "project_id": 456,
  "form_id": 123,
  "filters": [
    {
      "field": "field3",
      "operator": "gt",
      "value": 10
```

```json
    }
  ],
  "group_by": ["field2"],
  "aggregations": [
    {
      "field": "field1",
      "functions": ["sum", "mean", "count"]
    }
  ],
  "sort": [
    {
      "field": "field1_mean",
      "order": "desc"
    }
  ],
  "options": {
    "include_visualizations": true
  }
}
```

Response:

```json
{
  "success": true,
  "data": {
    "exploration_id": "exploration_123",
    "results": [
      {
        "field2": "B",
        "field1_sum": 31.5,
        "field1_mean": 15.75,
        "field1_count": 2
      },
      {
        "field2": "A",
        "field1_sum": 18.3,
        "field1_mean": 18.3,
        "field1_count": 1
      },
      {
        "field2": "C",
        "field1_sum": 10.5,
        "field1_mean": 10.5,
        "field1_count": 1
      }
    ],
    "visualizations": [
      {
        "type": "bar_chart",
        "x": "field2",
        "y": "field1_mean",
        "url": "https://example.com/visualizations/
bar_chart_field1_mean_by_field2.png"
      }
    ]
  },
  "meta": {
    "timestamp": "2023-06-12T17:20:15Z"
  }
}
```

Report API

The Report API allows you to generate and manage reports.

Create Report

Endpoint: `POST /api/v1/data-aggregation/reports`

Request Body:

```
{
  "project_id": 456,
  "title": "Project Analysis Report",
  "description": "A comprehensive analysis of project data",
  "sections": [
    {
      "title": "Summary",
      "content": "This report provides a summary of the project
data."
    },
    {
      "title": "Descriptive Statistics",
      "analysis_id": "analysis_123"
    },
    {
      "title": "Inferential Statistics",
      "analysis_id": "analysis_124"
    },
    {
      "title": "Data Exploration",
      "exploration_id": "exploration_123"
    }
  ],
  "format": "pdf",
  "options": {
    "include_cover_page": true,
    "include_table_of_contents": true,
    "include_executive_summary": true
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "report_id": "report_123",
    "title": "Project Analysis Report",
    "created_at": "2023-06-12T17:25:45Z",
    "status": "PROCESSING",
    "estimated_completion": "2023-06-12T17:30:45Z"
  },
  "meta": {
    "timestamp": "2023-06-12T17:25:45Z"
  }
}
```

Get Report

Endpoint: `GET /api/v1/data-aggregation/reports/{report_id}`

Path Parameters:

- **report_id**: ID of the report to retrieve

Response:

```
{
  "success": true,
  "data": {
    "report": {
      "id": "report_123",
      "project_id": 456,
      "title": "Project Analysis Report",
      "description": "A comprehensive analysis of project data",
      "created_at": "2023-06-12T17:25:45Z",
      "completed_at": "2023-06-12T17:30:45Z",
      "status": "COMPLETED",
      "url": "https://example.com/reports/report_123.pdf",
      "format": "pdf",
      "size": 1024000
    }
  },
  "meta": {
    "timestamp": "2023-06-12T17:35:22Z"
  }
}
```

List Reports

Endpoint: `GET /api/v1/data-aggregation/reports`

Query Parameters:

- **project_id**: (Optional) Filter reports by project ID
- **status**: (Optional) Filter reports by status
- **page**: (Optional) Page number for pagination (default: 1)
- **per_page**: (Optional) Number of items per page (default: 10)
- **sort_by**: (Optional) Field to sort by (default: "created_at")
- **sort_order**: (Optional) Sort order ("asc" or "desc", default: "desc")

Response:

```json
{
  "success": true,
  "data": {
    "reports": [
      {
        "id": "report_123",
        "project_id": 456,
        "title": "Project Analysis Report",
        "created_at": "2023-06-12T17:25:45Z",
        "status": "COMPLETED",
        "format": "pdf"
      },
      ...
    ]
  },
  "meta": {
    "page": 1,
    "per_page": 10,
    "total_pages": 1,
    "total_items": 5
  }
}
```

Delete Report

Endpoint: `DELETE /api/v1/data-aggregation/reports/{report_id}`

Path Parameters:

- **report_id**: ID of the report to delete

Response:

```
{
  "success": true,
  "data": {
    "message": "Report deleted successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T17:40:30Z"
  }
}
```

Integration API

The Integration API allows you to integrate the system with external services.

Baserow Integration

Configure Baserow

Endpoint: `POST /api/v1/data-aggregation/integrations/baserow/configure`

Request Body:

```
{
  "url": "https://baserow.example.com",
  "api_token": "your_baserow_api_token",
  "enabled": true
}
```

Response:

```
{
  "success": true,
  "data": {
    "integration_id": "baserow_integration",
    "url": "https://baserow.example.com",
    "enabled": true,
    "status": "CONNECTED",
    "configured_at": "2023-06-12T17:45:45Z"
  },
  "meta": {
    "timestamp": "2023-06-12T17:45:45Z"
  }
}
```

Get Baserow Configuration

Endpoint: `GET /api/v1/data-aggregation/integrations/baserow`

Response:

```json
{
  "success": true,
  "data": {
    "integration": {
      "id": "baserow_integration",
      "url": "https://baserow.example.com",
      "enabled": true,
      "status": "CONNECTED",
      "configured_at": "2023-06-12T17:45:45Z",
      "last_sync": "2023-06-12T17:45:45Z"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T17:50:22Z"
  }
}
```

Sync with Baserow

Endpoint: `POST /api/v1/data-aggregation/integrations/baserow/sync`

Request Body:

```json
{
  "project_id": 456,
  "form_id": 123,
  "table_id": "baserow_table_id",
  "field_mapping": {
    "field1": "baserow_field1",
    "field2": "baserow_field2"
  }
}
```

Response:

```json
{
  "success": true,
  "data": {
    "sync_id": "sync_123",
    "project_id": 456,
    "form_id": 123,
    "table_id": "baserow_table_id",
    "status": "COMPLETED",
    "records_synced": 27,
    "synced_at": "2023-06-12T17:55:30Z"
  },
  "meta": {
    "timestamp": "2023-06-12T17:55:30Z"
  }
}
```

AI Tool Integration

Configure AI Tool

Endpoint: `POST /api/v1/data-aggregation/integrations/ai-tool/configure`

Request Body:

```json
{
  "tool": "claude",
  "api_key": "your_claude_api_key",
  "model": "claude-3-opus-20240229",
  "enabled": true
}
```

Response:

```json
{
  "success": true,
  "data": {
    "integration_id": "claude_integration",
    "tool": "claude",
    "model": "claude-3-opus-20240229",
    "enabled": true,
    "status": "CONNECTED",
    "configured_at": "2023-06-12T18:00:45Z"
  },
  "meta": {
    "timestamp": "2023-06-12T18:00:45Z"
    }
  }
```

Get AI Tool Configuration

Endpoint: `GET /api/v1/data-aggregation/integrations/ai-tool`

Response:

```json
{
  "success": true,
  "data": {
    "integration": {
      "id": "claude_integration",
      "tool": "claude",
      "model": "claude-3-opus-20240229",
      "enabled": true,
      "status": "CONNECTED",
      "configured_at": "2023-06-12T18:00:45Z"
    }
  },
  "meta": {
    "timestamp": "2023-06-12T18:05:22Z"
  }
}
```

Generate AI Analysis

Endpoint: `POST /api/v1/data-aggregation/integrations/ai-tool/analyze`

Request Body:

```json
{
  "project_id": 456,
  "form_id": 123,
  "prompt": "Analyze the relationship between field1 and
field2",
  "data_filter": {
    "field3": {
      "operator": "gt",
      "value": 10
    }
  }
}
```

Response:

```json
{
  "success": true,
  "data": {
    "analysis_id": "ai_analysis_123",
    "project_id": 456,
    "form_id": 123,
    "status": "COMPLETED",
    "result": {
      "text": "The analysis shows a strong positive correlation
between field1 and field2...",
      "visualizations": [
        {
          "type": "scatter_plot",
          "url": "https://example.com/visualizations/
scatter_plot_field1_field2.png"
        }
      ]
    },
    "completed_at": "2023-06-12T18:10:30Z"
  },
  "meta": {
    "timestamp": "2023-06-12T18:10:30Z"
  }
}
```

Webhooks

The ODK MCP System supports webhooks for event notifications.

Register Webhook

Endpoint: `POST /api/v1/data-aggregation/webhooks`

Request Body:

```json
{
  "url": "https://your-server.com/webhook",
  "events": ["submission.created", "form.updated"],
  "project_id": 456,
  "secret": "your_webhook_secret"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "webhook_id": "webhook_123",
    "url": "https://your-server.com/webhook",
    "events": ["submission.created", "form.updated"],
    "project_id": 456,
    "created_at": "2023-06-12T18:15:45Z"
  },
  "meta": {
    "timestamp": "2023-06-12T18:15:45Z"
  }
}
```

List Webhooks

Endpoint: `GET /api/v1/data-aggregation/webhooks`

Response:

```
{
  "success": true,
  "data": {
    "webhooks": [
      {
        "id": "webhook_123",
        "url": "https://your-server.com/webhook",
        "events": ["submission.created", "form.updated"],
        "project_id": 456,
        "created_at": "2023-06-12T18:15:45Z"
      },
      ...
    ]
  },
  "meta": {
    "timestamp": "2023-06-12T18:20:22Z"
  }
}
```

Delete Webhook

Endpoint: `DELETE /api/v1/data-aggregation/webhooks/{webhook_id}`

Path Parameters:

- **webhook_id**: ID of the webhook to delete

Response:

```
{
  "success": true,
  "data": {
    "message": "Webhook deleted successfully"
  },
  "meta": {
    "timestamp": "2023-06-12T18:25:30Z"
  }
}
```

Webhook Payload

When an event occurs, the system sends a POST request to the registered webhook URL with the following payload:

```
{
  "event": "submission.created",
  "timestamp": "2023-06-12T18:30:45Z",
  "project_id": 456,
  "data": {
    "submission_id": 789,
    "form_id": 123,
    "submitted_by": "user",
    "submitted_at": "2023-06-12T18:30:45Z"
  },
  "signature": "computed_signature"
}
```

The signature is computed using HMAC-SHA256 with the webhook secret:

```
signature = HMAC-SHA256(webhook_secret, event + timestamp +
JSON.stringify(data))
```

Rate Limiting

The API implements rate limiting to prevent abuse. The rate limits are:

- **Authentication endpoints**: 10 requests per minute per IP address
- **Other endpoints**: 60 requests per minute per authenticated user

When a rate limit is exceeded, the API returns a 429 Too Many Requests response with the following body:

```json
{
  "success": false,
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Rate limit exceeded",
    "details": {
      "limit": 60,
      "remaining": 0,
      "reset": 1623517845
    }
  },
  "meta": {
    "timestamp": "2023-06-12T18:35:45Z"
  }
}
```

The response also includes the following headers:

- **X-RateLimit-Limit**: The rate limit (requests per minute)
- **X-RateLimit-Remaining**: The number of requests remaining in the current window
- **X-RateLimit-Reset**: The time at which the current rate limit window resets (Unix timestamp)

Versioning

The API is versioned using the URL path. The current version is v1:

```
/api/v1/...
```

When a new version is released, it will be available at a new path:

```
/api/v2/...
```

The API follows semantic versioning:

- **Major version** (v1, v2): Breaking changes
- **Minor version** (v1.1, v1.2): New features, no breaking changes
- **Patch version** (v1.1.1, v1.1.2): Bug fixes, no breaking changes

The current API version can be retrieved using the following endpoint:

Endpoint: `GET /api/version`

Response:

```json
{
  "success": true,
  "data": {
    "version": "1.0.0",
    "released_at": "2023-06-01T00:00:00Z"
  },
  "meta": {
    "timestamp": "2023-06-12T18:40:22Z"
  }
}
```

Examples

Authentication and Form Creation

```python
import requests
import json

# Base URLs
base_url = "http://localhost:8000"
form_management_url = f"{base_url}/api/v1/form-management"
data_aggregation_url = f"{base_url}/api/v1/data-aggregation"

# Authenticate
auth_response = requests.post(
    f"{data_aggregation_url}/auth/login",
    json={
        "username": "admin",
        "password": "password"
    }
)
auth_data = auth_response.json()
token = auth_data["data"]["token"]

# Create a project
project_response = requests.post(
    f"{data_aggregation_url}/projects",
    json={
        "name": "Example Project",
        "description": "A project for example purposes"
    },
    headers={
        "Authorization": f"Bearer {token}"
    }
)
project_data = project_response.json()
project_id = project_data["data"]["project_id"]
```

```python
# Upload an XLSForm
with open("example_form.xlsx", "rb") as f:
    form_response = requests.post(
        f"{form_management_url}/forms/upload",
        data={
            "name": "Example Form",
            "project_id": project_id
        },
        files={
            "xlsform": ("example_form.xlsx", f)
        },
        headers={
            "Authorization": f"Bearer {token}"
        }
    )
form_data = form_response.json()
form_id = form_data["data"]["form_id"]

print(f"Created form with ID: {form_id}")
```

Data Collection and Analysis

```python
 import requests
import json

# Base URLs
base_url = "http://localhost:8000"
data_collection_url = f"{base_url}/api/v1/data-collection"
data_aggregation_url = f"{base_url}/api/v1/data-aggregation"

# Authenticate
auth_response = requests.post(
    f"{data_aggregation_url}/auth/login",
    json={
        "username": "admin",
        "password": "password"
    }
)
auth_data = auth_response.json()
token = auth_data["data"]["token"]

# Submit data
submission_response = requests.post(
    f"{data_collection_url}/submissions",
    json={
        "form_id": 123,
        "project_id": 456,
        "data": {
            "field1": 15,
            "field2": "B",
                "field3": 20
            }
        },
        headers={
            "Authorization": f"Bearer {token}"
        }
    )
    submission_data = submission_response.json()
    submission_id = submission_data["data"]["submission_id"]

    # Perform descriptive analysis
    analysis_response = requests.post(
        f"{data_aggregation_url}/analysis/descriptive",
        json={
            "project_id": 456,
            "form_id": 123,
            "variables": ["field1", "field2", "field3"],
            "options": {
                "include_summary": True,
                "include_frequency": True,
                "include_visualizations": True
            }
        },
        headers={
            "Authorization": f"Bearer {token}"
        }
    )
    analysis_data = analysis_response.json()
    analysis_id = analysis_data["data"]["analysis_id"]

    # Generate a report
    report_response = requests.post(
        f"{data_aggregation_url}/reports",
        json={
            "project_id": 456,
            "title": "Data Analysis Report",
            "description": "A report of the collected data",
            "sections": [
                {
                    "title": "Descriptive Statistics",
                    "analysis_id": analysis_id
                }
            ],
            "format": "pdf",
            "options": {
                "include_cover_page": True
            }
        },
        headers={
            "Authorization": f"Bearer {token}"
        }
    )
```

```
report_data = report_response.json()
report_id = report_data["data"]["report_id"]

# Check report status
report_status_response = requests.get(
    f"{data_aggregation_url}/reports/{report_id}",
    headers={
        "Authorization": f"Bearer {token}"
    }
)
report_status_data = report_status_response.json()
report_url = report_status_data["data"]["report"]["url"]

print(f"Report available at: {report_url}")
```

References

1. Open Data Kit Documentation
2. XLSForm Standard
3. JSON Web Token (JWT) Standard
4. HTTP Status Codes
5. RESTful API Design Best Practices
6. OpenAPI Specification
7. Baserow API Documentation
8. Webhook Best Practices
9. Rate Limiting Best Practices
10. API Versioning Best Practices