# Partner AI Agent Integration Requirements Gathering

## 1. Overview

This document outlines some of the concepts for TurboDocx's AI Agents for retrieving data within your platform. We show a mock REST API Design and Database schema, and explain how TurboDocx traverses your APIs.

## 2. API Requirements

- **Authentication**: For proof of concepts, we prefer API Key authentication, but we can support OAuth at a later point.
- **Data Format**: JSON over HTTP.
- **Rate Limits**: Please provide example documentation on rate limit throttles.
- **Error Handling**: Partners should provide documentation on their error handling mechanisms.

## 3. Partner REST API Design Example

### 3.1 Endpoints

| Method | Endpoint | Description |
|---|---|---|
| GET | /projects | Fetch all projects |
| GET | /projects/{id} | Fetch a single project by ID |
| GET | /contacts/{id} | Fetch a single contact by ID |
| GET | /deals/{id} | Fetch a single deal by ID |

## 3.2 UML Table Schema Mapping Example

**Projects Table**

| Field | Type | Notes |
|---|---|---|
| id | UUID | Primary Key |
| name | String | Project name |
| description | Text | Project details |
| contact_id | UUID | Foreign Key to Contacts |
| deal_id | UUID | Foreign Key to Deals |
| created_at | Timestamp | Auto-generated |
| updated_at | Timestamp | Auto-updated |

**Contacts Table**

| Field | Type | Notes |
|---|---|---|
| id | UUID | Primary Key |
| name | String | Contact name |
| email | String | Contact email |
| phone | String | Contact phone |

**Deals Table**

| Field | Type | Notes |
|---|---|---|
| id | UUID | Primary Key |
| name | String | Deal name |
| amount | Decimal | Deal value |
| status | String | Deal status |

### 3.3 Example: Fetching a Project

**Note:** This section is an example of how TurboDocx likes to integrate to get a full context of the data the user is requesting for our AI to process the information.

**Scenario:** Suppose we are integrating with a CRM and need to fetch project details for "Project Alpha."

**Request:**

```
GET /projects/123e4567-e89b-12d3-a456-426614174000
```

**Response:**

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "name": "Project Alpha",
  "description": "CRM implementation for Client X",
  "created_at": "2024-02-27T10:00:00Z",
  "updated_at": "2024-02-27T12:00:00Z"
}
```

### 3.4 Example: Fetching Related Records Recursively

**Note:** This section is an example of how TurboDocx likes to integrate to get a full context of the data the user is requesting for our AI to process the information.

**Scenario:** Fetch all related contacts and deals for "Project Alpha" by making separate calls using IDs from previous responses.

**Step 1: Fetch Project Details**

**Request:**

```
GET /projects/123e4567-e89b-12d3-a456-426614174000
```

**Response:**

```
{
  "id": "123e4567-e89b-12d3-a456-426614174000",
  "name": "Project Alpha",
  "description": "CRM implementation for Client X",
  "contact_id": "987e4567-e89b-12d3-a456-426614174000",
  "deal_id": "654e4567-e89b-12d3-a456-426614174000"
}
```

**Step 2: Fetch Contact Details**

**Note:** Using the `contact_id` from the previous API call.

**Request:**

```
GET /contacts/987e4567-e89b-12d3-a456-426614174000
```

**Response:**

```
{
  "id": "987e4567-e89b-12d3-a456-426614174000",
  "name": "John Doe",
  "email": "john.doe@example.com",
  "phone": "+1-555-1234"
}
```

**Step 3: Fetch Deal Details**

**Note:** Using the `deal_id` from the previous API call.

**Request:**

```
GET /deals/654e4567-e89b-12d3-a456-426614174000
```

**Response:**


```
{
  "id": "654e4567-e89b-12d3-a456-426614174000",
  "name": "Enterprise Software Sale",
  "amount": 50000,
  "status": "Closed-Won"
}
```


**Step 4: (On the TurboDocx side, we will aggregate this data into a structured format for AI processing, as shown below)**

**Note:** When working with AI, human-readable JSON improves performance. The clearer the data, the smarter the AI.


```
{
  "project": {
    "id": "123e4567-e89b-12d3-a456-426614174000",
    "name": "Project Alpha",
    "description": "CRM implementation for Client X",
    "contact": {
      "id": "987e4567-e89b-12d3-a456-426614174000",
      "name": "John Doe",
      "email": "john.doe@example.com",
      "phone": "+1-555-1234"
    },
    "deal": {
      "id": "654e4567-e89b-12d3-a456-426614174000",
      "name": "Enterprise Software Sale",
      "amount": 50000,
      "status": "Closed-Won"
    }
  }
}
```

## 4. Key Mapping and Documentation

If any keys in the API responses are unclear or not self-explanatory, **Partners should provide documentation** mapping API fields to friendly names and where they appear in the Partner's UI. This helps our AI identify what is important.

### Example Mapping:

| API Key | UI Display Name |
|---|---|
| project_val | Project Name |
| contact_person | Primary Contact Name |
| deal_amount | Total Deal Value |

## 5. Batch Endpoints

Does your API support batch operations? If so, please provide details on how they work, including:

- Available batch operations (e.g., fetching multiple projects at once, bulk updates)
- Request and response format
- Rate limits for batch processing
- Any specific considerations for batch operations