LOCATION TREE API DOCUMENTATION

Author: Tmone Nguyen (vthinh.nguyen@gmail.com)

Company: Surbana Technologies

Version: 1.0.0

Date: April 18, 2025

PURPOSE

The Location Tree API is a hierarchical location management system designed to help facility managers, building administrators, and property developers efficiently organize and manage spatial data within buildings. This system allows for precise tracking of spaces, from entire buildings down to individual rooms, with accurate area measurements and logical organization reflecting the physical structure of real-world buildings.

INTRODUCTION

The Location Tree API is a RESTful service that provides a hierarchical location management system. It allows for creating, reading, updating, and deleting locations in a tree structure, with support for parent-child relationships. This documentation describes all available endpoints and operations.

API BASE URL

The base URL for all API endpoints is:

`http://localhost:3000/api`

A Swagger UI documentation interface is also available at:

`http://localhost:3000/api/docs`

LOCATION ENTITY

The core entity in this API is a Location, which has the following properties:

- **id**: Unique identifier (UUID)
- **name**: Name of the location
- **locationNumber**: Unique location number (e.g., LOC-001)

- **area**: Area size of the location (numeric value)
- **parentId**: ID of the parent location (null for root locations)
- **level**: Hierarchy level of the location (0 for root locations, increments by 1 for each level down)
- **path**: String representation of the location's path in the hierarchy (e.g., "parent_id.location_id")
- **createdAt**: Timestamp of when the record was created
- **updatedAt**: Timestamp of when the record was last updated

API ENDPOINTS

1. Create a New Location

HTTP Method: POST

URL: /api/locations

Description: Creates a new location in the system. If a parentId is provided, the new location will be created as a child of that parent location.

Request Body:

```
json
{
  "name": "Warehouse A",
  "locationNumber": "LOC-001",
  "area": 1500.75,
  "parentId": "optional-parent-uuid"
}
```

Response: Returns the created location object with all properties including the generated ID, path, and level.

Status Codes:

- 201: Location created successfully
- 400: Invalid input data
- 404: Parent location not found (if parentId was provided)

2. Get All Locations

HTTP Method: GET

URL: /api/locations

Description: Returns a list of all locations in the system, ordered by path to maintain hierarchical order.

Response: An array of location objects.

Status Codes:

- 200: Locations found

3. Get Location Tree

HTTP Method: GET

URL: /api/locations/tree

Description: Returns a hierarchical tree structure of all locations, with each location containing its children in a nested format.

Response: An array of root location objects, each containing nested children objects.

Status Codes:

- 200: Location tree generated successfully

4. Get Children of a Location

HTTP Method: GET

URL: /api/locations/children/

Description

: Returns all immediate child locations for a specific parent location.

URL Parameters:

- id: UUID of the parent location

Response: An array of location objects that are direct children of the specified parent.

Status Codes:

- 200: Children locations found
- 404: Parent location not found

5. Get a Specific Location

HTTP Method: GET

URL: /api/locations/

Description

: Returns a specific location by its ID.

URL Parameters:

- id: UUID of the location to retrieve

Response: A single location object.

Status Codes:

- 200: Location found
- 404: Location not found

6. Update a Location

HTTP Method: PATCH

URL: /api/locations/

Description

: Updates a specific location. When changing the parent of a location, the API automatically updates the level and path of the location and all its descendants.

URL Parameters:

- id: UUID of the location to update

Request Body: Partial location object with properties to update.

```json
{
  "name": "Updated Warehouse",
  "area": 1800.50,
  "parentId": "new-parent-uuid"
}
```

Response: The updated location object.

Status Codes:

- 200: Location updated successfully
- 400: Invalid input data or attempt to create a cycle in the hierarchy
- 404: Location not found or new parent not found

7. Delete a Location

HTTP Method: DELETE

URL: /api/locations/

Description

: Deletes a specific location. The operation will fail if the location has children.

URL Parameters:

- id: UUID of the location to delete

Response: No content.

Status Codes:

- 200: Location deleted successfully
- 400: Cannot delete a location that has children
- 404: Location not found

EXAMPLE USAGE

1. Creating a Root Location

POST /api/locations

Request Body:

```json
{
  "name": "Building A",
  "locationNumber": "BLDG-A",
  "area": 5000
}
```

Response:

```json
{
  "id": "30edc3d0-64a8-428f-a77e-b44cb148f9bb",
  "name": "Building A",
  "locationNumber": "BLDG-A",
  "area": 5000,
  "parentId": null,
  "level": 0,
  "path": "30edc3d0-64a8-428f-a77e-b44cb148f9bb",
  "createdAt": "2025-04-18T08:00:00.000Z",
  "updatedAt": "2025-04-18T08:00:00.000Z"
}
```

2. Creating a Child Location

POST /api/locations

Request Body:

json

```
{
  "name": "First Floor",
  "locationNumber": "BLDG-A-F1",
  "area": 1500,
  "parentId": "30edc3d0-64a8-428f-a77e-b44cb148f9bb"
}
```

Response:

json

```
{
  "id": "c7723614-325f-4142-a8c9-b020af25e49b",
  "name": "First Floor",
  "locationNumber": "BLDG-A-F1",
  "area": 1500,
  "parentId": "30edc3d0-64a8-428f-a77e-b44cb148f9bb",
  "level": 1,
  "path": "30edc3d0-64a8-428f-a77e-b44cb148f9bb.c7723614-325f-4142-a8c9-b020af25e49b",
  "createdAt": "2025-04-18T08:05:00.000Z",
  "updatedAt": "2025-04-18T08:05:00.000Z"
}
```

3. Moving a Location to a Different Parent

PATCH /api/locations/c7723614-325f-4142-a8c9-b020af25e49b

Request Body:

json

```
{
  "parentId": "5b7e87b2-4daf-46c2-b3a5-a1e1986872b0"
}
```

Response:

```json
{
  "id": "c7723614-325f-4142-a8c9-b020af25e49b",
  "name": "First Floor",
  "locationNumber": "BLDG-A-F1",
  "area": 1500,
  "parentId": "5b7e87b2-4daf-46c2-b3a5-a1e1986872b0",
  "level": 1,
  "path": "5b7e87b2-4daf-46c2-b3a5-a1e1986872b0.c7723614-325f-4142-a8c9-b020af25e49b",
  "createdAt": "2025-04-18T08:05:00.000Z",
  "updatedAt": "2025-04-18T09:15:00.000Z"
}
```

NOTES AND RESTRICTIONS

- When creating or updating locations, the API automatically manages the level and path properties
- You cannot make a location its own parent or create cycles in the location hierarchy
- You cannot delete a location that has children; you must delete the children first or move them elsewhere
- The API uses UUID format for all IDs
- Location paths use dot-notation to represent the hierarchy (e.g., "parent_id.child_id.grandchild_id")
- The API follows RESTful conventions and supports standard HTTP methods

ERROR HANDLING

All errors follow a standard format:

```json
{
  "statusCode": 404,
  "message": "Location with ID 123e4567-e89b-12d3-a456-426614174000 not found",
  "error": "Not Found"
}
```

Common errors include:

- 400 Bad Request: Invalid input data or validation errors
- 404 Not Found: Requested resource doesn't exist
- 403 Forbidden: Cannot perform the requested operation (e.g., deleting a location with children)

CONCLUSION

The Location Tree API provides a complete solution for managing hierarchical location data with built-in support for parent-child relationships and tree-based operations. The API automatically manages the integrity of the location hierarchy, ensuring consistency across operations. For more detailed information, refer to the Swagger documentation available through the API URL.

CONTACT INFORMATION

For questions or support, please contact:

- Email: support@surbana.tech
- Technical Contact: luc.le@coe.surbana.tech