USER MANUAL

Universal Hierarchy Viewer

A configurable Lightning Web Component for displaying and managing hierarchical Salesforce data

VERSION	AUTHOR	API VERSION	LAST UPDATED
3.4	Bart Debruyne	63.0	April 2026

Table of Contents

1 Introduction

The **Universal Hierarchy Viewer** is a powerful and flexible Lightning Web Component (LWC) designed to display and manage hierarchical data within Salesforce. It allows administrators to configure dynamic tree-views for any combination of standard and custom objects, supporting features like lazy loading, search, drag-and-drop reparenting, and custom context menu actions.

This manual provides a comprehensive guide for configuring and utilizing the sfdcbdb Hierarchy Viewer to meet your specific business needs.

2 Key Features

Flexible Hierarchy Display

Supports Direct Mode (single root) and Starting Layer Mode (multiple roots from a parent record) for maximum flexibility.

Tree & Drag-and-Drop Views

Toggle between a classic expandable tree table and an intuitive drag-and-drop card interface for reparenting records.

Lazy Loading

Optimize performance for large hierarchies by loading child nodes only when expanded, or load the full hierarchy upfront.

Up to 6 Dynamic Columns

Configure custom columns to display relevant fields, including relationship fields (e.g., `RecordType.Name`).

Client-Side Search with Highlighting

Filter hierarchy nodes by search term and see matching text highlighted in yellow directly within the cells, similar to browser Ctrl+F.

Custom Context Menu

Define right-click actions using Custom Metadata: navigate, open URL, launch Flow, or execute Apex.

Inline Column Editing

Make one column editable directly in the hierarchy. Supports text, number, date, picklist, and more. Includes pencil icon, yellow dirty-cell highlight, and a global Save button.

HTML Formula Field Rendering

Formula fields that output HTML (e.g., `<img>` tags for color indicators) are rendered as rich content instead of raw text.

Undo Last Move

Quickly revert the most recent drag-and-drop operation with a single click.

Permission-Based Drag & Drop

Control access to drag-and-drop reparenting via the `sfdcbdb_TreeViewDragAndDrop` custom permission.

Hierarchy Statistics: The component displays real-time statistics on the number of records and depth levels in the loaded hierarchy, visible in the header bar.

3 Use Cases

The sfdcbdb Hierarchy Viewer is ideal for scenarios where you need to visualize and manage related records in a hierarchical structure:

Asset Management

View a parent-child hierarchy of Assets, allowing field service technicians or inventory managers to see all components of a larger asset and move them between locations or parent assets via drag & drop.

Work Order & Line Item Management

Display Work Orders and their nested Work Order Line Items, enabling dispatchers or service managers to manage tasks within a work order structure.

Account & Contact / Opportunity Hierarchy

Visualize complex Account hierarchies with associated Contacts or Opportunities, providing a clear overview of customer relationships.

Location Management

Manage a hierarchy of physical locations, allowing administrators to organize sites, buildings, and rooms.

Product Structure

Model product assemblies where products have sub-components, facilitating manufacturing or inventory planning.

User / Role Hierarchy

Display manager-subordinate relationships within your Salesforce org.

Inline Field Updates

Allow users to update a specific field (e.g., Status, Priority, Serial Number) directly in the hierarchy without navigating to each record. Ideal for bulk data maintenance on hierarchical structures.

4 Component Placement

The component can be added to the following Salesforce page types:

- Lightning Record Pages
- Experience Cloud Pages
- Lightning Flow Screens

> **How to add:** In Lightning App Builder, open the component panel on the left and search for **sfdcbdb Hierarchy Viewer**. Drag it onto the desired region of your page layout. Once placed, configure the properties in the right-hand panel.

5 Configuration Properties

The following properties are available when adding the component to a Lightning Page or Flow Screen via App Builder or Flow Builder.

5.1 General Configuration

PROPERTY	TYPE	DEFAULT	DESCRIPTION
`componentTitle`	String	`Hierarchy Viewer`	The title displayed in the component header.
`recordId` *(Flow only)*	String	—	The ID of the record to display the hierarchy for. Automatically passed on Record Pages. **Required for Flows.**

5.2 Core Object Configuration

These properties define the main objects involved in your hierarchy.

PROPERTY	REQUIRED	DEFAULT	DESCRIPTION
`parentObjectApiName`	Yes	`Account`	API name of the parent object in the hierarchy (e.g., `WorkOrder`, `Location`).
`childObjectApiName`	Yes	`Asset`	API name of the child object (e.g., `WorkOrderLineItem`, `Asset`). Can be the same as `parentObjectApiName` for self-referencing hierarchies.
`parentFieldApiName`	Yes	AccountId	Lookup field on the **child object** that references its parent (e.g., `WorkOrderId` on `WorkOrderLineItem`).
`childRelationshipName`	Yes	Assets	Child relationship name from parent to children (e.g., `WorkOrderLineItems`, `Assets`).

5.3 Advanced: Starting Layer Configuration

Use these properties when your hierarchy needs to start from a record that sits "above" your main parent object. Eg show all Location records of the account.

PROPERTY	DEFAULT	DESCRIPTION
`startingObjectApiName`	Account	Optional. API name of a starting object layer (e.g., `Account`). When specified, the component queries all parent object records associated with this starting record.
`startingToParentFieldApiName`	AccountId	Optional. Lookup field on the parent object that points to the starting object (e.g., `AccountId` on `Location`). Required when `startingObjectApiName` is set.
`startingToParentRelationshipName`	Locations	Optional. Child relationship name from starting object to parent objects.

> **Example:** To display Locations and their Assets under an Account, set `startingObjectApiName` to `Account`, `parentObjectApiName` to `Location`, `childObjectApiName` to `Asset`, and `startingToParentFieldApiName` to `AccountId`.

5.4 Logic Configuration

PROPERTY	DEFAULT	DESCRIPTION
`parentSelfReferenceFieldApiName`	—	Lookup field for self-referencing relationships on the parent object (e.g., `ParentLocation__c` on `Location`, `ParentId` on `Account`). Enables nested hierarchies of the same parent type.
`childSelfReferenceFieldApiName`	—	Lookup field for self-referencing relationships on the child object (e.g., `ParentId` on `Asset`). Enables nested hierarchies of the same child type.
`additionalFields`	—	Comma-separated list of extra fields to query. Included in server-side query and client-side search but **not displayed as columns**. Useful for enabling search on non-display fields.

5.5 UI Configuration

PROPERTY	TYPE	DEFAULT	DESCRIPTION
`autoExpandOnLoad`	Boolean	`false`	When enabled, all nodes expand on initial load. Disable for better performance with large hierarchies.
`lazyLoad`	Boolean	`false`	If `true`, only top-level nodes load initially; children are fetched on demand when expanded. Can be toggled at runtime via the UI.
`showDetailsButton`	Boolean	`true`	Shows an "Info" button next to each record, opening a detail panel for the selected item.
`contextMenuModalSize`	String	`xlarge`	Default modal size for Flow actions. Valid values: `large`, `xlarge`, `full`. Overridden by `Modal_Size__c` on individual action records.
`editableColumnIndex`	Integer	`0`	Sets one column (2–6) as inline-editable. Set to `0` to disable. When enabled, a pencil icon appears next to editable cells, and a global "Save Changes" button appears when changes are pending.

5.6 Column Configuration (Up to 6 Columns)

Each column requires a **label** and a **field API name**. Column 1 is always visible and serves as the primary identifier.

> **Important:** For objects without a standard `Name` field (e.g., `WorkOrder`), use their primary identifier field for Column 1. For example: `WorkOrderNumber` for WorkOrder, `LineItemNumber` for WorkOrderLineItem, `CaseNumber` for Case.

PROPERTY	REQUIRED	DEFAULT	DESCRIPTION
`column1Label` / `column1Field`	Yes	`Name`	Label and field API name for the first column (primary identifier). Supports relationship fields.
`column2Label` / `column2Field`	No	—	Label and field API name for the second column (e.g., `RecordType.Name`).
`column3Label` / `column3Field`	No	—	Label and field API name for the third column.

PROPERTY	REQUIRED	DEFAULT	DESCRIPTION
`column4Label` / `column4Field`	No	—	Label and field API name for the fourth column.
`column5Label` / `column5Field`	No	—	Label and field API name for the fifth column.
`column6Label` / `column6Field`	No	—	Label and field API name for the sixth column.

5.7 Drag & Drop Card Configuration

Configure up to 2 additional fields displayed on each card in the Drag & Drop view.

PROPERTY	DEFAULT	DESCRIPTION
`dragDropField1Label` / `dragDropField1`	—	Label and API name for the first extra field on drag-and-drop cards (e.g., `Status`). Leave blank to hide.
`dragDropField2Label` / `dragDropField2`	—	Label and API name for the second extra field (e.g., `SerialNumber`). Leave blank to hide.

Note: Relationship fields (e.g., `RecordType.Name`) are **not supported** on drag-and-drop cards. Only direct fields on the object are supported.

Example — Asset Drag & Drop: To enable reparenting of Assets under other Assets within an Account hierarchy, configure `parentObjectApiName` = `Account`, `childObjectApiName` = `Asset`, `parentFieldApiName` = `AccountId`, `childRelationshipName` = `Assets`, and `childSelfReferenceFieldApiName` = `ParentId`. Users can then drag an Asset onto another Asset to reparent it, or drag it back to the Account level to remove the parent relationship.

Important: Drag-and-drop reparenting requires that the parent lookup field (`parentFieldApiName`) is editable. Objects like `WorkOrder` and `WorkOrderLineItem` have non-editable parent fields (e.g., `WorkOrderId`), meaning you **cannot** move a line item from one Work Order to another. However, rearranging the child-to-child hierarchy within the same parent (via `childSelfReferenceFieldApiName`, e.g., `ParentWorkOrderLineItemId`) is still supported.

5.8 Inline Column Editing

The component supports making **one column** (columns 2–6) editable directly within the hierarchy grid. This allows users to modify field values without leaving the hierarchy view.

How It Works

1. Set `editableColumnIndex` to a value between `2` and `6` in the component properties (App Builder or Flow Builder).
2. A **pencil icon** appears next to each cell in the selected column for all rows where the field is updateable.
3. Click the pencil icon to enter edit mode. The cell becomes an input field (or a picklist dropdown for picklist fields) and automatically receives focus.
4. Modify the value. The cell is highlighted in **yellow** to indicate an unsaved change.
5. A global **"Save Changes"** button appears at the bottom. Click it to persist all pending changes to Salesforce in a single transaction.

Supported Field Types

The data type of the editable field is automatically detected from the Salesforce schema. The component renders the appropriate input control:

SALESFORCE FIELD TYPE	RENDERED AS
Text, Text Area	Text input
Number, Currency, Percent	Number input
Date	Date picker
Date/Time	DateTime picker
Email	Email input
Phone	Phone input
URL	URL input
Checkbox	Toggle

SALESFORCE FIELD TYPE	RENDERED AS
Picklist	Combobox dropdown with active picklist values

> **Example — Editable Industry Column:** To allow users to edit the `Industry` field directly in the Account hierarchy, configure the component with `column2Label` = `Industry`, `column2Field` = `Industry`, and `editableColumnIndex` = `2`. Since Industry is a picklist, clicking the pencil icon will show a dropdown with all active picklist values.

> **Example — Editable Serial Number on Assets:** To let users update the `SerialNumber` on Assets within a hierarchy, configure `column3Label` = `Serial Number`, `column3Field` = `SerialNumber`, and `editableColumnIndex` = `3`. This renders a text input for direct inline editing.

> **Security:** Inline editing respects Salesforce CRUD and Field-Level Security (FLS). The pencil icon only appears for fields that the current user has update permission on. All updates are executed with `AccessLevel.USER_MODE` to enforce object and field-level access.

> **Limitations:** Only **one column** can be editable at a time. Column 1 (the Name column) cannot be made editable because it serves as the tree navigation column with expand/collapse controls. Relationship fields (e.g., `RecordType.Name`) and formula fields are read-only and will not display the pencil icon.

5.9 HTML Formula Field Rendering

When a column displays a formula field that outputs HTML content (e.g., an `<img>` tag for color indicators), the component automatically detects this and renders the HTML as rich content instead of displaying it as raw text.

> **Example — Color Indicator:** If you have a formula field `Color_Indicator__c` on Asset that outputs `<img src="/img/samples/light_green.gif" alt="Green"/>`, configure it as a column (e.g., `column4Field` = `Color_Indicator__c`). The component will render the actual color image instead of showing the raw HTML tag.

5.10 Search Substring Highlighting

When a user types in the search box, the component filters the hierarchy to show only matching nodes (and their ancestors). Additionally, the **matching portion of text** within each cell is highlighted with a yellow background, making it easy to spot exactly where the search term appears.

> **How it works:** The search scans all configured columns (1–6) plus any `additionalFields`. Matching substrings in column cells are wrapped in a highlight, similar to a browser's Ctrl+F. If a match is found only in `additionalFields` (not in a visible column), a search indicator icon appears next to the row.

> **Example:** If the hierarchy displays Accounts with an `Industry` column and the user searches for `enter`, all rows containing that substring are shown. The text "Enterprise" in the Industry column would display as "Enterprise" with "**enter**" highlighted in yellow.

> **Note:** Search highlighting applies to plain text and picklist columns. Date/DateTime columns and HTML formula fields are excluded from text highlighting because they use specialized renderers.

5.11 Date & DateTime Locale Formatting

Columns displaying **Date** or **DateTime** fields are automatically formatted according to the running user's locale settings. Instead of showing raw ISO format (e.g., `2026-04-14`), dates are displayed in the user's regional format (e.g., `04/14/2026` for US, `14/04/2026` for EU).

> **Example:** Configure `column5Label` = `Last Visit Date` and `column5Field` = `Last_Visit_Date__c`. The component detects the field's data type as `DATE` and automatically renders it using the user's locale. DateTime fields additionally display hours and minutes.

6 Context Menu Actions

The component supports a configurable context menu (right-click) on each hierarchy node. Actions are defined using the `sfdcbdb_Hierarchy_Context_Action__mdt` Custom Metadata Type.

6.1 Available Action Types

ACTION TYPE	BEHAVIOR	REQUIRED FIELD
Navigate	Navigates to the record's standard detail page (SPA navigation).	—
Url	Opens a URL template in a new tab. Use `{recordId}` as a placeholder for the clicked record's ID.	`Url_Template__c`
Flow	Launches a Salesforce Screen Flow in a modal. The flow receives the `recordId` as an input variable.	`Flow_Api_Name__c`
Apex	Executes an Apex class that implements the `Callable` interface. Receives `Map<String, Object>{ 'recordId' ⇒ recordId }` as input.	`Apex_Class__c`

6.2 Custom Metadata Fields

FIELD	REQUIRED	DESCRIPTION
`Label__c`	Yes	Display label for the action in the context menu.
`Action_Type__c`	Yes	Type of action: `Navigate`, `Url`, `Flow`, or `Apex`.
`Object_Api_Name__c`	No	Restrict action to a specific object. Leave blank to apply to **all objects**.
`Sequence__c`	No	Controls display order (lower numbers appear first).
`Icon_Name__c`	No	SLDS icon name (e.g., `utility:info`, `standard:account`).
`Modal_Size__c`	No	Modal size for Flow actions: `large`, `xlarge`, `full`.

FIELD	REQUIRED	DESCRIPTION
`Apex_Class__c`	For Apex	Fully qualified Apex class name (must implement `Callable`).
`Flow_Api_Name__c`	For Flow	API name of the Screen Flow to launch.
`Url_Template__c`	For Url	URL template with `{recordId}` placeholder.

6.3 Auto-Refresh After Apex Actions

After a context menu **Apex** action completes successfully, the hierarchy automatically refreshes to reflect any data changes made by the Apex class. No manual refresh is needed.

6.4 How to Configure

1. In Salesforce Setup, navigate to **Custom Metadata Types**.
2. Find **sfdcbdb Hierarchy Context Action** and click **Manage Records**.
3. Click **New** and fill in the fields according to the desired action type.

Example — Apex Color Update: To create a context menu action that updates the color of all child Assets recursively, create a Custom Metadata record with `Label__c` = `Update Colors`, `Action_Type__c` = `Apex`, `Apex_Class__c` = `UpdateAssetHierarchyColor`, `Object_Api_Name__c` = `Asset`, and `Icon_Name__c` = `utility:color_swatch`. The Apex class must implement `Callable` and be declared as `global`.

Important — Apex Class Visibility: When the Hierarchy Viewer is installed as a **managed package**, any custom Apex class called from the context menu in the **subscriber org** must be declared as `global with sharing` (not `public`). This is required because the managed package uses `Type.forName()` to dynamically instantiate the class, and only `global` classes are visible across namespace boundaries.

Example — Callable Apex Class Skeleton:

```
global with sharing class MyCustomAction implements Callable {
    public Object call(String action, Map<String, Object> args) {
        Id recordId = (Id) args.get('recordId');
```

```
        // Your business logic here
        return null;
    }
}
```

7 Limitations

Maximum Hierarchy Depth

40 levels

The component limits iterative loading of same-type nested records (e.g., Asset under Asset) to 40 levels. If your hierarchy exceeds this depth, deeper levels will not be displayed. A notification badge is shown in the header when this limit is reached.

Maximum Records Per Query

10,000 records

Individual SOQL queries are capped at 10,000 records. Extremely wide hierarchies may be truncated.

Query Batch Size

5,000 IDs per batch

When querying children, parent IDs are processed in batches of 5,000. This helps manage Salesforce governor limits for large datasets.

Maximum Display Columns

6 columns

The tree table supports up to 6 configurable data columns.

Drag & Drop Card Fields

2 fields max

Each drag-and-drop card supports up to 2 additional display fields. Relationship fields (e.g., `RecordType.Name`) are not supported on cards.

Auto-Expand Performance

For very large or deep hierarchies, enabling `autoExpandOnLoad` can impact initial loading performance. Consider using `lazyLoad` instead.

Objects Without a Standard Name Field

Objects like `WorkOrder` and `WorkOrderLineItem` do not have a standard `Name` field. The administrator must configure `column1Field` with the correct primary identifier (e.g., `WorkOrderNumber`, `LineItemNumber`).

Drag & Drop Access Control

The drag-and-drop view is only available to users who have been assigned the `sfdcbdb_TreeViewDragAndDrop` custom permission.

Inline Editing

1 column, columns 2–6 only

Only one column can be made editable at a time. Column 1 (the tree name column) cannot be made editable. Relationship fields and formula fields are read-only. Picklist fields render as a dropdown with active values from the schema.

Inline Editing — CRUD/FLS Enforcement

The pencil icon only appears for fields where the running user has update access. All save operations use `AccessLevel.USER_MODE` and validate field editability server-side before persisting changes.

8 Disclaimer

This component was developed by **Bart Debruyne**. It is provided "as is" without warranty of any kind, express or implied. The author disclaims all liability for any damages, errors, or issues arising from its use. Use at your own risk.