| **Back To Browsing** |
| --- |
|  |

| **Breadcrumbs** |
| --- |
| [Documentation](/en/docs/)[AEM as a Cloud Service](/en/docs/experience-manager-cloud-service)[User Guide](/en/docs/experience-manager-cloud-service/content/overview/introduction) |

**Model Definitions, Fields, and Component Types field-types**

| **Article Metadata** |
| --- |
| Last update: Thu Aug 14 2025 00:00:00 GMT+0000 (Coordinated Universal Time) |

| **Article Metadata Appliesto** |
| --- |
| * Applies to: * Experience Manager as a Cloud Service |

| **Article Metadata Topics** |
| --- |
| * Topics: * Developing |

| **Article Metadata Createdby** |
| --- |
| CREATED FOR:   * Admin * Developer |

Learn about fields and the component types that the Universal Editor can edit in the properties panel with examples. Understand how you can instrument your own app by creating a model definition and linking to the component.

**Overview overview**

When adapting your own apps for use with the Universal Editor, you must instrument the components and define what fields and component types they can manipulate in the properties panel of the editor. You do this by creating a model and linking to that from the component.

This document provides an overview of a model definition and of fields and the component types available to you along with example configurations.

| **Note (tip)** |
| --- |
| TIP |
| If you are not familiar with how to instrument your app for the Universal Editor, please see the document [Universal Editor Overview for AEM Developers](/en/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/developer-overview). |

**Model Definition Structure model-structure**

In order to configure a component via the properties panel in the Universal Editor, a model definition has to exist and be linked to the component.

The model definition is a JSON structure, starting with an array of models.

| **Code (language json)** |
| --- |
| [  {  "id": "model-id", // must be unique  "fields": [] // array of fields which shall be rendered in the properties panel  }  ] |

See the [**Fields**](#fields) section of this document for more information about how to define your fields array.

You can link a model to a component in two ways: using the [component definition](#component-definition) or [via the instrumentation.](#instrumentation)

---

**Linking Using the Component Definition component-definition**

This is the preferred method of linking the model to the component. Doing so allows you to maintain the link centrally in the component definition and enables dragging components across containers.

Simply include the model property in the component object in the components array in the component-definition.json file.

For details, please see the document [Component Definition.](/en/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/component-definition)

**Linking Using Instrumentation instrumentation**

To use the model definition with a component, the data-aue-model attribute can be used.

| **Code (language html)** |
| --- |
| <div data-aue-resource="urn:datasource:/content/path" data-aue-type="component" data-aue-model="model-id">Click me</div> |

| **Note** |
| --- |
| NOTE |
| The Universal Editor first checks if a model is linked via the instrumentation and uses that before checking the component definition. This means: |
| * Projects that have implemented the link to the model via the instrumentation will continue to work as-is with no need for change. * If you define the model in the [component definition](#component-definition) as well as in the instrumentation, the instrumentation will always be used. |

**Loading a Model Definition loading-model**

Once a model is created, it can be referenced as an external file.

| **Code (language html)** |
| --- |
| <script type="application/vnd.adobe.aue.model+json" src="<url-of-model-definition>"></script> |

Alternatively you can also define the model inline.

| **Code (language html)** |
| --- |
| <script type="application/vnd.adobe.aue.model+json">  { ... model definition ... }  </script> |

---

**Fields fields**

A field object has the following type definition.

| **Table (0 row 4, 1 row 4, 2 row 4, 3 row 4, 4 row 4, 5 row 4, 6 row 4, 7 row 4, 8 row 4, 9 row 4, 10 row 4, 11 row 4, 12 row 4, 13 row 4, 14 row 4)** | | | |
| --- | --- | --- | --- |
| Configuration | Value Type | Description | Required |
| component | ComponentType | Renderer of the component | Yes |
| name | string | Property [or path](#nesting) where the data shall be persisted | Yes |
| label | FieldLabel | Label of the field | Yes |
| description | FieldDescription | Description of the field | No |
| placeholder | string | Placeholder for the field | No |
| value | FieldValue | Default value | No |
| valueType | ValueType | Standard validation, can be string, string[], number, date, boolean | No |
| required | boolean | Is the field required | No |
| readOnly | boolean | Is the field read only | No |
| hidden | boolean | Is the field hidden by default | No |
| condition | RulesLogic | Rule to show or hide the field based on a [condition](/en/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/customizing#conditionally-hide) | No |
| multi | boolean | Is the field a multi field Note that container nesting is not permitted for multi-fields in the properties panel | No |
| validation | ValidationType | Validation rule or rules for the field | No |
| raw | unknown | Raw data which can be used by the component | No |

**name Field and Nesting nesting**

The name field can point directly to a property of the current resource, or in the case of components in cq:Pages, it can also use a path to a nested property. For example:

| **Code (language json)** |
| --- |
| "name": "teaser/image/fileReference" |

**Component Types component-types**

The following are the component types that are possible to use for rendering fields.

| **Table (0 row 2, 1 row 2, 2 row 2, 3 row 2, 4 row 2, 5 row 2, 6 row 2, 7 row 2, 8 row 2, 9 row 2, 10 row 2, 11 row 2, 12 row 2, 13 row 2, 14 row 2, 15 row 2, 16 row 2)** | |
| --- | --- |
| Description | Component Type |
| [AEM Tag](#aem-tag) | aem-tag |
| [AEM Content](#aem-content) | aem-content |
| [Boolean](#boolean) | boolean |
| [Checkbox Group](#checkbox-group) | checkbox-group |
| [Container](#container) | container |
| [Content Fragment](#content-fragment) | aem-content-fragment |
| [Date Time](#date-time) | date-time |
| [Experience Fragment](#experience-fragment) | aem-experience-fragment |
| [Multiselect](#multiselect) | multiselect |
| [Number](#number) | number |
| [Radio Group](#radio-group) | radio-group |
| [Reference](#reference) | reference |
| [Rich Text](#rich-text) | richtext |
| [Select](#select) | select |
| [Tab](#tab) | tab |
| [Text](#text) | text |

---

**AEM Tag aem-tag**

An AEM tag component type enables an AEM tag picker, which can be used to attach tags to the component.

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "aem-tag-picker",  "fields": [  {  "component": "aem-tag",  "label": "AEM Tag Picker",  "name": "cq:tags",  "valueType": "string"  }  ]  } | |
| Screenshot | Screenshot of AEM tag component type |

| **Note (tip)** |
| --- |
| TIP |
| Please see the document [Managing Taxonomy Data](https://www.aem.live/docs/authoring-taxonomy) for more information about how you can use spreadsheets to manage your taxonomy data for your Edge Delivery Services project. |

**AEM Content aem-content**

An AEM content component type enables an AEM content picker, which can be used to select any AEM resource. Unlike the [reference component](#reference), which can only select assets, the AEM content component can reference any AEM content. It offers an additional validation type.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| rootPath | string | Path that the content picker will open for the user to select AEM content, limiting selection to that directory and subdirectories | No |

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "aem-content-picker",  "fields": [  {  "component": "aem-content",  "name": "reference",  "value": "",  "label": "AEM Content Picker",  "valueType": "string",  "validation": {  "rootPath": "/content/refresh"  }  }  ]  } | |
| Screenshot | Screenshot of AEM content component type |

---

**Boolean boolean**

A boolean component type stores a simple true/false value rendered as a toggle. It offers an additional validation type.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| customErrorMsg | string | Message that will display if the value entered isn’t a boolean value | No |

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | {  "id": "boolean",  "fields": [  {  "component": "boolean",  "label": "Boolean",  "name": "boolean",  "valueType": "boolean"  }  ]  } | |
| Sample 2 | | **code language-json** | | --- | | {  "id": "another-boolean",  "fields": [  {  "component": "boolean",  "label": "Boolean",  "name": "boolean",  "valueType": "boolean",  "validation": {  "customErrorMsg": "Think, McFly. Think!"  }  }  ]  } | |
| Screenshot | Screenshot of boolean component type |

**Checkbox Group checkbox-group**

Similar to a boolean, a checkbox group component type allows for the selection of multiple true/false items, rendered as multiple checkboxes.

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "checkbox-group",  "fields": [  {  "component": "checkbox-group",  "label": "Checkbox Group",  "name": "checkbox",  "valueType": "string[]",  "options": [  { "name": "Option 1", "value": "option1" },  { "name": "Option 2", "value": "option2" }  ]  }  ]  } | |
| Screenshot | Screenshot of checkbox group component type |

---

**Container container**

A container component type allows the grouping of components including multifield support. It offers an additional configuration. Note that container nesting is not permitted for multi-fields in the properties panel

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Configuration | Value Type | Description | Required |
| collapsible | boolean | Is the container collapsible | No |

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "container",  "fields": [  {  "component": "container",  "label": "Container",  "name": "container",  "valueType": "string",  "collapsible": true,  "fields": [  {  "component": "text-input",  "label": "Simple Text 1",  "name": "text",  "valueType": "string"  },  {  "component": "text-input",  "label": "Simple Text 2",  "name": "text2",  "valueType": "string"  }  ]  }  ]  } | |
| Screenshot | Screenshot of container component type |
| Multifield Support | | **code language-json** | | --- | | {  "component": "container",  "name": "test",  "label": "Multi Text",  "multi": true,  "fields": [  {  "component": "reference",  "name": "image",  "value": "",  "label": "Sample Image",  "valueType": "string"  },  {  "component": "text",  "name": "alt",  "value": "",  "label": "Alt Text",  "valueType": "string"  }  ]  } | |

---

**Content Fragment content-fragment**

The Content Fragment picker can be used to select a [Content Fragment](/en/docs/experience-manager-cloud-service/content/sites/authoring/fragments/content-fragments) and its variations (if required). It offers an additional configuration.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Configuration | Value Type | Description | Required |
| variationName | string | Variable name to store the selected variation. If undefined, no variation picker is displayed | No |

It also offers an additional validation type.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| rootPath | string | Path that the content picker will open for the user to select the Content Fragment, limiting selection to that directory and subdirectories | No |

| **Note** |
| --- |
| NOTE |
| The Universal Editor [validates Content Fragment fields based on their models](/en/docs/experience-manager-cloud-service/content/assets/content-fragments/content-fragments-models#validation) allowing you to enforce data integrity rules such as regex patterns and uniqueness constraints. |
| This ensures that your content meets specific business requirements before it’s published. |

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | [  {  "id": "aem-content-fragment",  "fields": [  {  "component": "aem-content-fragment",  "name": "picker",  "label": "Content Fragment Picker",  "valueType": "string",  "variationName": "contentFragmentVariation",  "validation": {  "rootPath": "/content/refresh"  }  }  ]  }  ] | |
| Screenshot | Screenshot of Content Fragment picker |

**Date Time date-time**

A date time component type allows the specification of a date, time, or combination thereof. It offers additional configurations.

| **Table (0 row 4, 1 row 4, 2 row 4)** | | | |
| --- | --- | --- | --- |
| Configuration | Value Type | Description | Required |
| displayFormat | string | Format with which to display the date string | Yes |
| valueFormat | string | Format in which to store the date string | Yes |

It also offers an additional validation type.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| customErrorMsg | string | Message that will display if valueFormat isn’t met | No |

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | {  "id": "date-time",  "fields": [  {  "component": "date-time",  "label": "Date & Time",  "name": "date",  "valueType": "date"  }  ]  } | |
| Sample 2 | | **code language-json** | | --- | | {  "id": "another-date-time",  "fields": [  {  "component": "date-time",  "valueType": "date-time",  "name": "field1",  "label": "Date Time",  "description": "This is a date time field that stores both date and time.",  "required": true,  "placeholder": "YYYY-MM-DD HH:mm:ss",  "displayFormat": null,  "valueFormat": null,  "validation": {  "customErrorMsg": "Marty! You have to come back with me!"  }  },  {  "component": "date-time",  "valueType": "date",  "name": "field2",  "label": "Another Date Time",  "description": "This is another date time field that only stores the date.",  "required": true,  "placeholder": "YYYY-MM-DD",  "displayFormat": null,  "valueFormat": null,  "validation": {  "customErrorMsg": "Back to the future!"  }  },  {  "component": "date-time",  "valueType": "time",  "name": "field3",  "label": "Yet Another Date Time",  "description": "This is another date time field that only stores the time.",  "required": true,  "placeholder": "HH:mm:ss",  "displayFormat": null,  "valueFormat": null,  "validation": {  "customErrorMsg": "Great Scott!"  }  }  ]  } | |
| Screenshot | Screenshot of date time component type |

---

**Experience Fragment experience-fragment**

The Experience Fragment picker can be used to select an [Experience Fragment](/en/docs/experience-manager-cloud-service/content/sites/authoring/fragments/experience-fragments) and its variations (if required). It offers an additional configuration.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Configuration | Value Type | Description | Required |
| variationName | string | Variable name to store the selected variation. If undefined, no variation picker is displayed | No |

It also offers an additional validation type.

| **Table (0 row 4, 1 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| rootPath | string | Path that the content picker will open for the user to select the Experience Fragment, limiting selection to that directory and subdirectories | No |

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | [  {  "id": "experience-fragment",  "fields": [  {  "component": "aem-experience-fragment",  "valueType": "string",  "name": "experience-fragment",  "label": "experience-fragment",  "variationName": "experienceFragmentVariation",  "validation": {  "rootPath": "/content/refresh"  }  }  ]  }  ] | |
| Screenshot | Screenshot of Experience Fragment picker |

**Multiselect multiselect**

A multiselect component type presents multiple items for selection in a drop-down including the ability to group the selectable elements.

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | {  "id": "multiselect",  "fields": [  {  "component": "multiselect",  "name": "multiselect",  "label": "Multi Select",  "valueType": "string",  "options": [  { "name": "Option 1", "value": "option1" },  { "name": "Option 2", "value": "option2" }  ]  }  ]  } | |
| Sample 2 | | **code language-json** | | --- | | {  "id": "multiselect-grouped",  "fields": [  {  "component": "multiselect",  "name": "property",  "label": "Multiselect field",  "valueType": "string",  "required": true,  "maxSize": 2,  "options": [  {  "name": "Theme",  "children": [  { "name": "Light", "value": "light" },  { "name": "Dark", "value": "dark" }  ]  },  {  "name": "Type",  "children": [  { "name": "Alpha", "value": "alpha" },  { "name": "Beta", "value": "beta" },  { "name": "Gamma", "value": "gamma" }  ]  }  ]  }  ]  } | |
| Screenshots | Screenshot of multiselect component type Screenshot of multiselect component type with grouping |

---

**Number number**

A number component type allows for the input of a number. It offers additional validation types.

| **Table (0 row 4, 1 row 4, 2 row 4, 3 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| numberMin | number | Minimum number allowed | No |
| numberMax | number | Maximum number allowed | No |
| customErrorMsg | string | Message that will display if numberMin or numberMax isn’t met | No |

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | {  "id": "number",  "fields": [  {  "component": "number",  "name": "number",  "label": "Number",  "valueType": "number",  "value": 0  }  ]  } | |
| Sample 2 | | **code language-json** | | --- | | {  "id": "another-number",  "fields": [  {  "component": "number",  "valueType": "number",  "name": "field1",  "label": "Number Field",  "description": "This is a number field.",  "required": true,  "placeholder": null,  "validation": {  "numberMin": 0,  "numberMax": 88,  "customErrorMsg": "You also need 1.21 gigawatts."  }  }  ]  } | |
| Screenshot | Screenshot of number component type |

**Radio Group radio-group**

A radio group component type allows for a mutually-exclusive selection from multiple options rendered as a group similar to a checkbox group.

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "radio-group",  "fields": [  {  "component": "radio-group",  "label": "Radio Group",  "name": "radio",  "valueType": "string",  "options": [  { "name": "Option 1", "value": "option1" },  { "name": "Option 2", "value": "option2" }  ]  }  ]  } | |
| Screenshot | Screenshot of radio group component type |

---

**Reference reference**

An reference component type enables an AEM asset picker, which can be used to select any AEM asset to reference. Unlike the [AEM content component](#aem-content), which can select any AEM resource, the reference component can only reference assets. It offers an additional validation type.

A reference component type allows for a reference to another data object from the current object.

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "reference",  "fields": [  {  "component": "reference",  "label": "Reference",  "name": "reference",  "valueType": "string"  }  ]  } | |
| Screenshot | Screenshot of reference component type |

**Rich Text rich-text**

Rich text allows for multi-line, rich text input.

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | {  "id": "richtext",  "fields": [  {  "component": "richtext",  "name": "rte",  "label": "Rich Text",  "valueType": "string"  }  ]  } | |
| Screenshot | Screenshot of text area component type |

**Select select**

A select component type allows for selection of a single option from a list of predefined options in a drop-down menu.

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "select",  "fields": [  {  "component": "select",  "label": "Select",  "name": "select",  "valueType": "string",  "options": [  { "name": "Option 1", "value": "option1" },  { "name": "Option 2", "value": "option2" }  ]  }  ]  } | |
| Screenshot | Screenshot of select component type |

---

**Tab tab**

A tab component type allows you to group other input fields together on multiple tabs to improve layout organization for the authors.

A tab definition can be thought of as a separator in the array of fields. Everything that comes after a tab will be placed on that tab until a new tab is encountered, whereafter the following items will be placed on the new tab.

If you wish to have items that appear above all tabs, they must be defined before any tabs.

| **Tabs** | |
| --- | --- |
| Sample | | **code language-json** | | --- | | {  "id": "tab",  "fields": [  {  "component": "tab",  "label": "Tab 1",  "name": "tab1"  },  {  "component": "text-input",  "label": "Text 1",  "name": "text1",  "valueType": "string"  },  {  "component": "tab",  "label": "Tab 2",  "name": "tab2"  },  {  "component": "text-input",  "label": "Text 2",  "name": "text2",  "valueType": "string"  }  ]  } | |
| Screenshot | Screenshot of tab component type |

**Text text**

Text allows for a single line of text input. It includes additional validation types.

| **Table (0 row 4, 1 row 4, 2 row 4, 3 row 4, 4 row 4)** | | | |
| --- | --- | --- | --- |
| Validation Type | Value Type | Description | Required |
| minLength | number | Minimum number of characters allowed | No |
| maxLength | number | Maximum number of characters allowed | No |
| regExp | string | Regular expression which the input text must match | No |
| customErrorMsg | string | Message that will display if minLength, maxLength, and/or regExp is/are violated | No |

| **Tabs** | |
| --- | --- |
| Sample 1 | | **code language-json** | | --- | | {  "id": "simpletext",  "fields": [  {  "component": "text",  "name": "text",  "label": "Simple Text",  "valueType": "string"  }  ]  } | |
| Sample 2 | | **code language-json** | | --- | | {  "id": "another simpletext",  "fields": [  {  "component": "text",  "name": "text",  "label": "Simple Text",  "valueType": "string",  "valueFormat": "regexp",  "description": "This is a text input with validation.",  "required": true,  "validation": {  "minLength": 1955,  "maxLength": 1985,  "regExp": "^foo:.\*",  "customErrorMsg": "Why don't you make like a tree and get outta here?"  }  }  ]  } | |
| Screenshot | Screenshot of text component type |

| **Target Insertion** | |
| --- | --- |
| recommendation-more-help |  |

---

| **Toc** |
| --- |
| fbcff2a9-b6fe-4574-b04a-21e75df764ab |

---

| **Doc Actions** |
| --- |
|  |

| **Mini Toc** |
| --- |
|  |

| **Metadata** | |
| --- | --- |
| Title | Model Definitions, Fields, and Component Types | Adobe Experience Manager |
| Description | Learn about fields and the component types that the Universal Editor can edit in the properties panel with examples. Understand how you can instrument your own app by creating a model definition and linking to the component. |
| Header Fragment | /en/global-fragments/header |
| Footer Fragment | /en/global-fragments/footer |
| Theme | docs |
| Id | fd72f8aa-c973-450c-bc36-f2ae49df0cbc |
| Coveo Solution | Experience Manager;Experience Manager|Experience Manager as a Cloud Service |
| Breadcrumb Title | User Guide |
| Breadcrumb Url | /docs/experience-manager-cloud-service/content/overview/introduction.html |
| Cloud | Experience Cloud |
| Coveo Content Type | Documentation |
| Exl Id | cb4567b8-ebec-477c-b7b9-53f25b533192 |
| Feature | Developing |
| Feature Set | Experience Manager Assets, Experience Manager Sites, Experience Manager, Experience Manager Forms, Experience Manager Cloud Manager, Experience Manager Screens |
| Git Commit | fb33696ae7987081c7ee4cd35a233f54aaaa4c65 |
| Git Commit File | bb149cd43158bfd1ceb43b04cc536c8c8291f968 |
| Git Edit | https://github.com/AdobeDocs/experience-manager-cloud-service.en/tree/main/help/implementing/universal-editor/field-types.md |
| Git Filename | /help/implementing/universal-editor/field-types.md |
| Git Issue | https://github.com/AdobeDocs/experience-manager-cloud-service.en/issues/new |
| Git Repo | https://github.com/AdobeDocs/experience-manager-cloud-service.en |
| Index | true |
| Landing Page Description | Learn about fields and the component types that the Universal Editor can edit in the properties panel with examples. Understand how you can instrument your own app by creating a model definition and linking to the component. |
| Last Update | Thu Aug 14 2025 00:00:00 GMT+0000 (Coordinated Universal Time) |
| Original Solution | Experience Manager, Experience Manager as a Cloud Service |
| Pipeline\_filename | /help/implementing/universal-editor/field-types.md |
| Pipeline\_internal | false |
| Product | Adobe Experience Manager |
| Publish Url | https://experienceleague.adobe.com/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/field-types.html?lang=en |
| Recommendations | noDisplay |
| Role | Admin,Developer |
| Solution | Experience Manager,Experience Manager as a Cloud Service |
| Solution Title | Experience Cloud |
| Sub Product | Adobe Experience Manager as a Cloud Service |
| Type | Documentation |
| User Guide Description | Learn how Experience Manager as a Cloud Service works and what the software can do for you. |
| User Guide Title | AEM as a Cloud Service |
| Version | Experience Manager as a Cloud Service |
| Build Date | Sat Aug 16 2025 16:36:32 GMT+0000 (Coordinated Universal Time) |
| Breadcrumbs | [{"title":"Documentation","url":"/docs/"},{"title":"AEM as a Cloud Service","url":"/docs/experience-manager-cloud-service.md?lang=en"},{"title":"User Guide","url":"/docs/experience-manager-cloud-service/content/overview/introduction.html?lang=en"},{"title":"Model Definitions, Fields, and Component Types","url":""}] |
| Theme Color | #EAEAEA |
| Google | notranslate |
| Msvalidate.01 | 4A06753A9B5F813EC3822353ABE2550E |