Adobe Experience Cloud

Adobe

WHITEPAPER

The hybrid architecture of Adobe Experience Manager.

How to author and deliver multichannel content.

Table of Contents

Executive summary.

The ways in which people digitally connect with the world continue to expand, ranging from traditional web browsers and mobile apps to the newest wearable tech, virtual reality, and IoT devices. This technological growth challenges marketers to produce consistent, engaging experiences across multiple channels. IT organizations must not only support this landscape of connected devices but also prepare for the future.

Adobe Experience Manager, the leading solution for content management, features a decoupled, modular architecture and provides extensible capabilities that empower marketers to build fantastic experiences for any channel.

Content management in Experience Manager decouples content and its management from its delivery channels, enabling you to reuse content across any channel quickly and easily.

Content delivery in Experience Manager leverages a platform approach, which Experience Manager Sites, Screens, Assets, Forms, and content services all build upon. This approach creates a standardized, reusable and extensible content delivery architecture that can target any channel.

The channel-agnostic content management and content delivery framework of Experience Manager combine to form a hybrid content management system (CMS) that is capable of owning one channel or supporting multiple channels.

This white paper explains the hybrid architecture of Adobe Experience Manager and how brands can leverage its powerful authoring and delivery mechanisms to engage customers with relevant content on the channel and device of their choice.

The hybrid architecture of Adobe Experience Manager.

How to author and deliver multichannel content.

Why a hybrid CMS?

Many CMSs fall into the category of either a traditional or headless CMS.

Traditional CMS architecture.

A traditional CMS manages and delivers content on a single technology stack, minimizing total cost of ownership with efficient system maintenance and training. It controls all of the templating and presentation logic and outputs fully formatted HTML. A traditional CMS typically provides an end-to-end solution for a limited number of channels, like websites, where it "owns the glass."

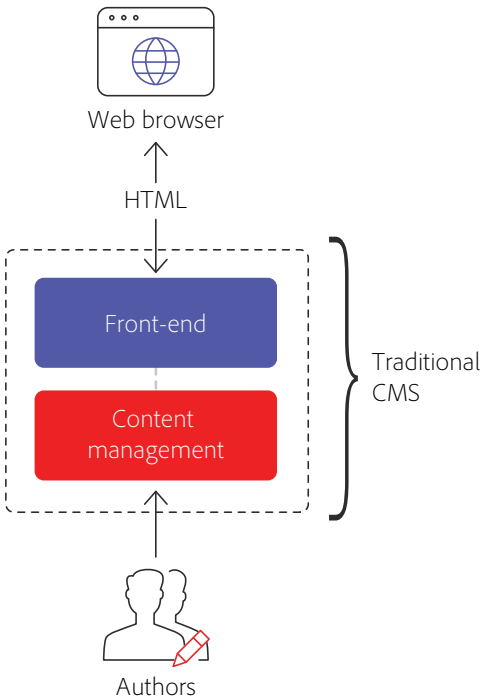


Figure 1. Traditional CMS architecture

Headless CMS architecture.

The term headless originates from the idea that the front-end presentation layer is the "head" of the application. Rather than delivering HTML or formatted content directly, a headless CMS decouples content from presentation, enabling content to be used by a variety of front-end technologies. A headless CMS exposes content as JSON (JavaScript Object Notation) through well-defined HTTP APIs. Headless CMSs are developer tools and lack business user functionality like preview, in-context editing and publishing workflows.

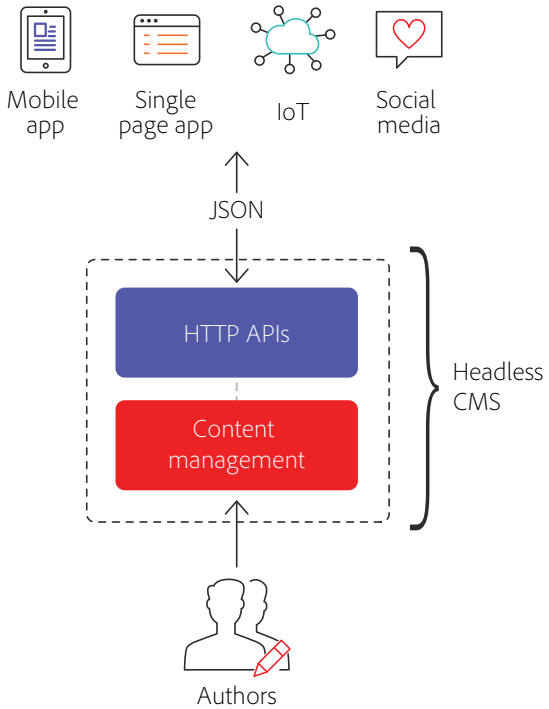


Figure 2. Typical headless architecture

A hybrid CMS combines the concepts of traditional and headless CMSs into a single architecture, resulting in the best of both worlds.

Traditional CMS advantages:	Headless CMS advantages:
• Enables marketers to offer a consistent message and personalized experiences • Empowers marketers to update presentation and layout with in-context previews • Lowers total cost of ownership with a single architecture stack and efficiencies in system maintenance and training	• Five Adobe-defined recovery scenarios, optimized to effectively address arising problems • Regular disaster recovery testing to ensure the effectiveness of recovery

Experience Manager: A hybrid CMS.

Experience Manager takes a hybrid approach that offers the best of both worlds: the efficiency and ease of use of a traditional CMS combined with the flexibility and scalability of a headless development framework. Experience Manager provides a central hub to house and distribute content and experiences to any channel and delivers content either as fully formatted HTML or JSON over HTTP APIs. With Experience Manager, brands can choose between traditional or headless delivery and can mix and match capabilities to meet their business needs.

Experience Manager offers a suite of capabilities that help brands deliver great digital experiences. With the following capabilities, Experience Manager acts like a traditional CMS and provides channel-centric experiences from end to end:

- Experience Manager Sites is the preeminent leader in experience and web content management (WCM), enabling enterprise and midmarket brands to design, author, manage, publish, and optimize their websites and single page applications (SPAs).
- Experience Manager Screens is a powerful tool for the digital signage market, allowing marketers to design, author, manage, schedule, deliver, and optimize dynamic content across digital displays.
- Experience Manager Forms transforms paper-based organizations through digital forms and customer communication solutions.

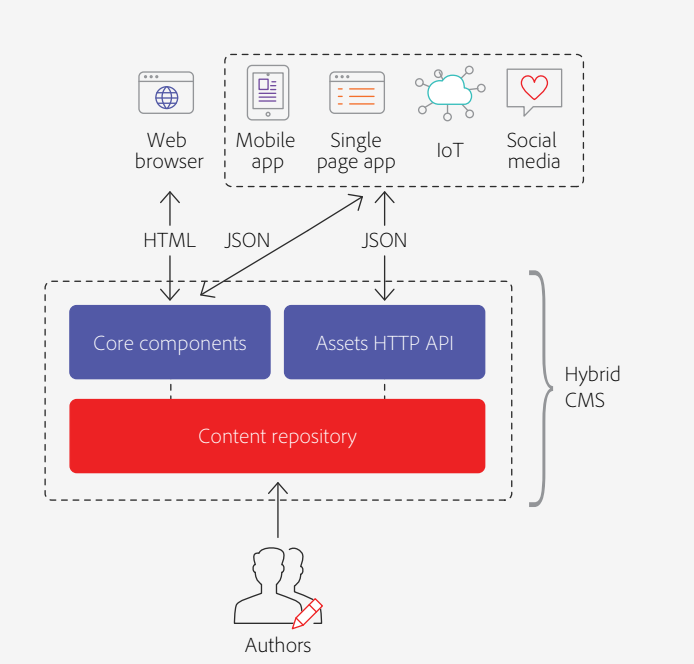


Figure 3. Adobe Experience Manager hybrid architecture

Recognizing that many other channels exist, Experience Manager includes features that express content in a variety of formats through API endpoints:

- **Content services** exposes content via two ways—direct content API or customizable API.
- **Sling Model Exporter** allows developers to quickly render any Experience Manager content into JSON using custom business logic.

Multichannel authoring with Experience Manager.

The hybrid architecture of Experience Manager enables brands to create content once and deliver to multiple channels. Authors create and manage content in a centralized location, improving efficiency and message consistency while maintaining the flexibility to optimize experiences per channel. Content can be an image, text-based editorial snippets, or a combination of several pieces of content that creates a reusable experience.

Content Fragment and Experience Fragment components in Experience Manager support these reusable omnichannel experiences:

Content fragments.

A content fragment is a design- and presentation-agnostic set of content that is destined to be used over and over again throughout a website or multiple channels and experiences. Content Fragments can contain unstructured data, for example, text and images, or structured data elements based on a data model. Unstructured Content Fragments are ideal for articles or other long forms of text, as authors can focus on writing and rely on downstream channels to manage layout and formatting. Structured content fragments are ideal for business-specific data , like speaker bios, FAQs, or product and feature descriptions. A text summarization feature enables authors to create variations of content optimized for downstream channels with varying screen sizes. Content Fragments can take advantage of version control, approval workflows, and translation services.

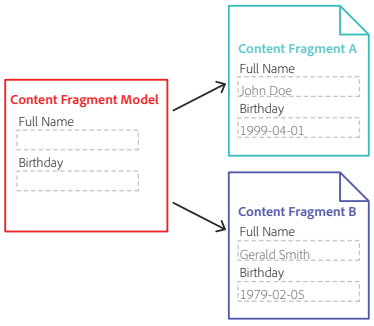


Figure 4. Example of a content fragment model

Content fragment models define the data schema for a content fragment and contain the following fields:

- Text
- Multiline text
- Number
- Boolean
- Date and time
- Enumeration
- Tags
- Reference

A drag-and-drop user interface empowers authors to generate content fragment models quickly and easily without coding, specifying data types, and adding data entry rules.

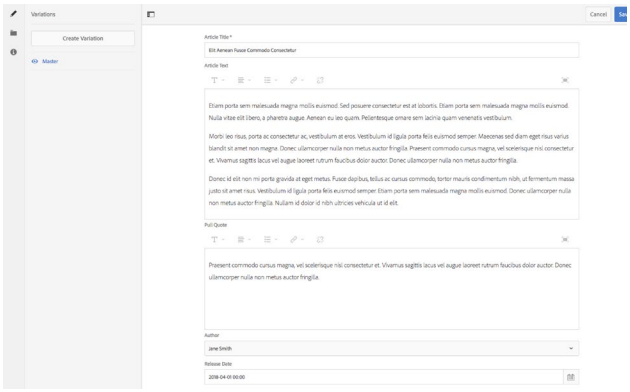


Figure 5. Authoring a content fragment in Experience Manager

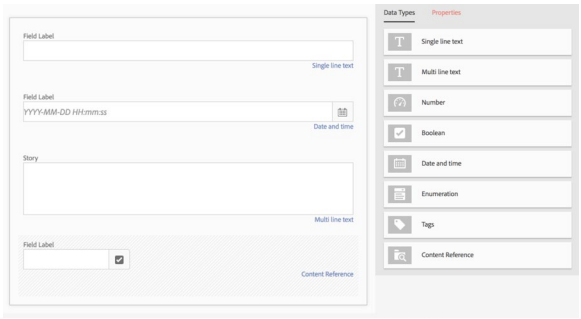


Figure 6. Screen showing how to create a content fragment

From a single content fragment, you can generate one or more variants that adhere to the model and then use the variants for different channels.

Use Content Fragments when the content:

- Is channel agnostic
- Adheres to a structure
- Does not require a specific layout or design
- Is not directly connected to the channels or experiences that expose it

Use Content Fragments for HTML delivery in:

- An Experience Manager Sites page to transform the content it contains into HTML that can be styled for display on web pages
- Experience Fragment to transform the content it contains into HTML that can be styled—for more information, read about Experience Fragments

Delivery mechanisms for Content Fragments.

- Combine Content Fragments with formatting and templating and deliver as fully formatted HTML
- Deliver as JSON with the Experience Manager assets HTTP API
- Use the Content Fragment List component to dynamically pull Content Fragments into a list based on a filter and number of desired fragments and deliver as JSON.
- Deliver in JSON with Experience Manager content services.

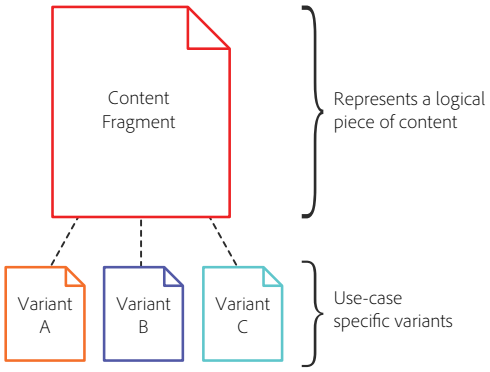


Figure 7. Use-case specific varients

Key considerations—Content Fragments:

- Leverage the features of Experience Manager, including metadata, workflow, versioning (Sites), discoverability, security, collaboration, and content intelligence (Assets)
- Leverage localization capabilities in Experience Manager
- Adhere to a defined data model, ensuring data consistency; data models can change over time, but changes affect all Content Fragments that use them
- Support long-form editorial text, such as an article, where the text is an element of the article's data model
- Are not directly accessed, but rather provide content to channels, which are responsible for managing the presentation of the content fragment
- Can create relationships with other pieces of content—for example, a content fragment can reference an image or even another content fragment; however, the reference is path-based, and if the referenced content moves, the path will no longer point to the correct location

Experience Fragments.

An Experience Fragment combines one or more pieces of content with design and layout. A good example of an Experience Fragment is a promotional experience composed of a banner image, text, and a call to action button. Experience Fragments allow marketers to manage experiences from a central location and ensure a consistent message while delivering contextually optimized content to each channel. Experience Fragments are standalone experiences designed to optimize display for different channels, such as a web page, social feed, or mobile app.

Variations of Experience Fragments, like Content Fragments can be created to address a different context or channel, optimizing the presentation of the core experience to best align with the channel and its audience.

Authors create variations using a predefined Experience Fragment template that supports any channel. Templates can be reused across Experience Fragments or even across variations of a single Experience Fragment, accelerating the rate of creation and publishing of channel-specific content. Most Experience Fragment variations are web based and intended for a browser. Experience Manager provides variations for social posting to Facebook and Pinterest out of the box.

Key considerations—Experience Fragments:

- Are built on Experience Manager templates and components, simplifying self-service configuration by authors
- Support content reuse between variations using the live copy feature, which provides intelligent content synchronization
- Support content reuse between variations as well as other Experience Fragments
- Provide in-context content creation, allowing authors to visualize how the experience will appear in other channels
- Contain presentation elements, so third-party integration is via HTML snippets
- Provide out-of-the box social media templates for Facebook and Pinterest platforms—support for additional social media platforms requires custom code

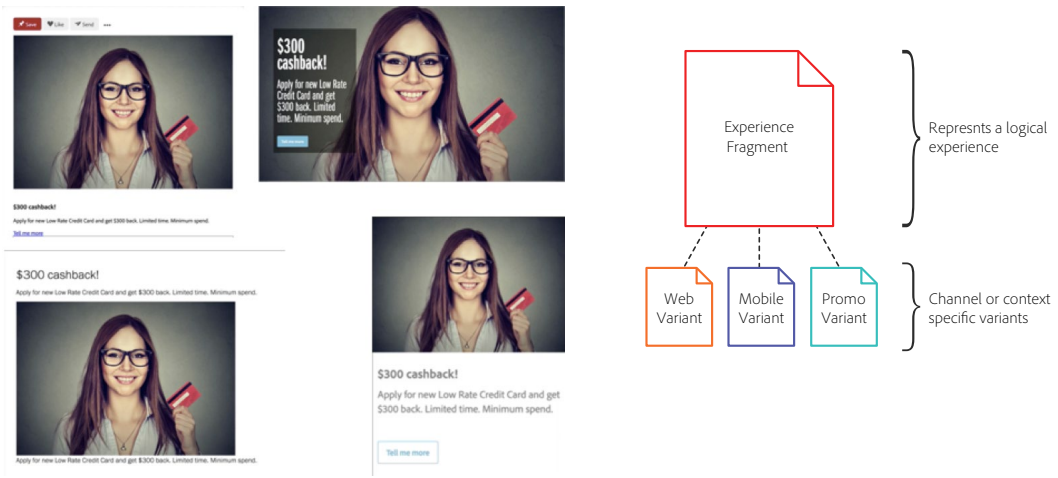


Figure 8. Example of an Experience Fragment viewed on different channels

Use Experience Fragments when the experience:

- Can stand on its own
- Displays differently across channels
- Includes layout or design

Use cases for Experience Fragments include:

- A website product promotion that is syndicated to third-party websites for cross-promotion
- A campaign synchronized across a brand's website, mobile app, and social media feeds
- A/B testing and audience targeting, enabled by out-of-the-box integration with Adobe Target, that optimizes Experience Fragment variations by channel or audience

Delivery mechanisms for Experience Fragments.

- Embed server-side with out-of-the-box components into other Experience Manager features like Sites or Screens and deliver as HTML.
- Embed variations into third-party applications over HTTP endpoints and integrate as HTML.
- Post variations to social media channels via respective social media APIs.
- Send variations to Adobe Target for dynamic embedding in a Sites or third-party website.

Single Page Application (SPA) Editor.

Single-page applications (SPAs) are growing in popularity as the performance and responsiveness of web experiences become critical. Developers can build these experiences rapidly with popular JavaScript frameworks such as React and Angular, typically using decoupled or "headless" CMSs. However, this approach cuts marketers and authors out of presentation or layout decisions. Time to market and total cost of ownership increase, as any presentation changes require development support.

Sites solves these problems while enabling developers to use their familiar frameworks and development tools. The Sites SPA Editor offers a WYSIWYG user interface that enables marketers and authors to make in-context changes to content, layout, and presentation, just as they would with traditional web pages. Any changes or updates are immediately reflected in the SPA.

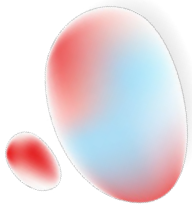
Key considerations—SPA Editor:

- Is intended for use with Sites
- Is powered by content services with JSON content from Sites
- Supports integration with React and Angular frameworks via an SDK
- Enables developers to continue to use familiar front-end technologies to develop the SPA
- Enables marketers to leverage familiar Experience Manager authoring tools to edit the SPA

Multichannel content delivery in Experience Manager.

To support all channels, CMSs must expose the content they manage in a multi-format and scalable way. There are two standard formats for exposing web content: HTML and JSON. HTML has long been the language of the web browser, while JSON is a lightweight delivery format that powers modern connected applications.

A robust CMS should provide features that maximize the reusability of content regardless of how it was originally created. Experience Manager templates, pages, and components are used to compose content which can be delivered via both HTML and JSON. It also provides HTTP APIs to pull raw content out of the content repository as JSON below the page level. Components can reference and reuse content, such as Content Fragments, Experience Fragments, and assets, within the same channel or across channels. The following sections will explore in detail how Experience Manager delivers content at and below the page level via HTML and JSON.



Delivering HTML content in Sites, Screens, and Forms.

Experience Manager can operate as a traditional web CMS for delivering HTML in Sites, Screens, and Forms. Templates define layout, content, and configurations and are responsible for including and supporting assets like CSS, JavaScript, and fonts. When used for web pages, templates also dictate which components authors can use on pages. A WYSIWYG, in-context authoring experience grants authors complete control over content layout and formatting. Components render in HTML and allow authors to configure, aggregate, or reference Experience Manager content or create new content inline.

Web Content Management (WCM) components

Using components, Experience Manager enables brands to reuse content across pages and channels and render the content as HTML at runtime. This structure promotes consistency across channels and offers a single, centralized location for updating content. However, authors also have the flexibility to modify components for a particular page. For example, authors can reference an image component stored in the DAM system, include it on a web page, and modify its properties for that page only.

Experience Manager includes a set of core components—such as breadcrumbs, lists, navigation, text, images, and more—that speed up website development. New projects should use core components as a starting point. To meet business requirements, developers can create custom components that extend core component functionality or implement entirely new functionality.

Authors drag and drop components and combine them with other components to form a web page. WCM Components are re-used but individually configured with different content.

Delivering content as JSON with Experience Manager content services framework.

Experience Manager allows for pure headless delivery of content below the page level with JSON output of resources directly from the Experience Manager Sites content repository. Content Fragments are also available for direct delivery as JSON using the assets HTTP API. The assets HTTP API allows for channel-agnostic content to be authored in-context OR without context and delivered headless anywhere as JSON. The current implementation of the assets HTTP API is REST and it allows for create-read-update-delete (CRUD) operations, including binary, metadata, renditions, and comments.

In addition, Experience Manager content services is a zero-code framework that exposes content in a standard JSON format. Content services allow brands to create dedicated HTTP endpoints separate from traditional Experience Manager channels. It serializes content via a JSON exporter, automatically exposing Experience Manager content, such as Content Fragments, in JSON. This promotes the reuse of Experience Manager content while decoupling the JSON HTTP endpoint from the rest of the application, giving front-end developers a consistent API to code against.

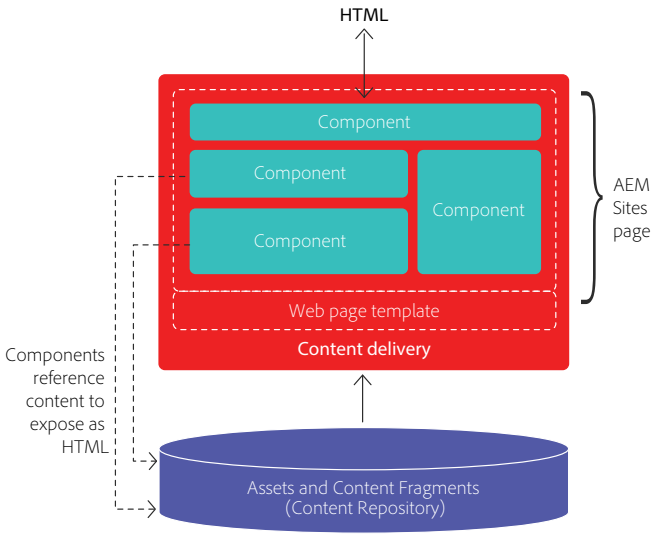


Figure 9. Delivering HTML with Experience Manager

Experience Manager content services builds on top of Apache Sling Model Exporter and provides its own JSON schema for components that implement its interfaces. Content services still leverages Experience Manager templates, pages, and components, but outputs to JSON instead of HTML.

- **Templates** define the top-level JSON schema and structure for downstream consumers by enforcing which components must be on a page and which components authors can add.
- **Pages** define the HTTP API endpoints that deliver JSON. Developers can define HTTP APIs using one or more pages, with each page acting as an HTTP endpoint that delivers a logical set of JSON data.
- **Components are** units of data that allow authors to choose and configure which data to expose. Similar to the HTML use case, authors can configure components to aggregate and reference other content in Experience Manager.

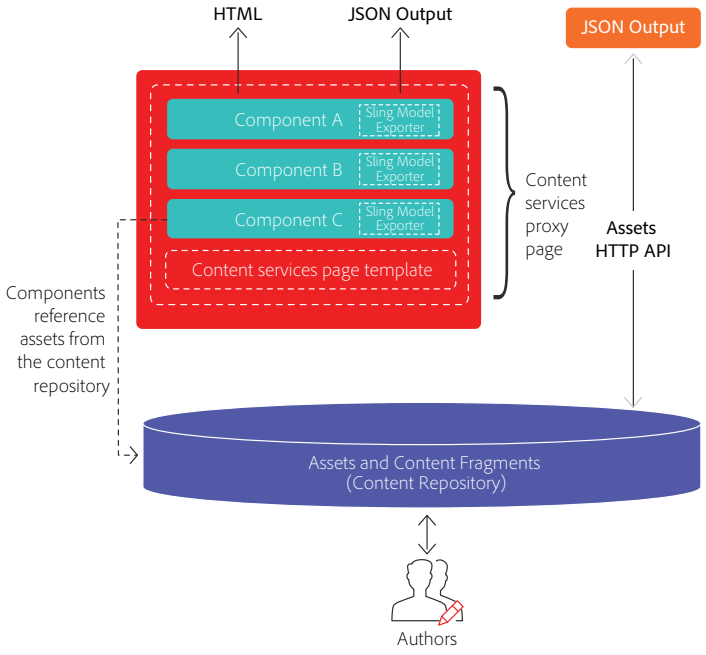


Figure 10. Delivering HTML or JSON with Experience Manager

Summary: HTML Delivery vs. JSON Delivery in Experience Manager.

	HTML delivery	JSON delivery
Capability or feature	Sites, Screens, Forms	Content services
Underlying technology	Sling HTTP web framework	Assets HTTP API or Sling Model Exporter
Templates	Responsive layouts that define the structure and appearance of a web page	JSON schema definitions
Pages	Traditional pages forming an Experience Manager website, based on templates and components	HTTP API JSON endpoints adhering to the JSON schema definition that includes templates and components
Components	Modules that collect and present content on a page	Modules that collect and transform content into normalized JSON
Referenced content	Content Fragments, Experience Fragments, Assets	Content Fragments, Assets

Best practices for multichannel content delivery

Implementing a CMS that serves all audiences and channels can seem like a daunting task. The following tips will help maximize success.

- **Start small and be agile.** Take a phased approach to building out your content management strategy, and measure your success before implementing larger projects. Remember to react and evolve—Rome wasn't built in a day.
- **Consider future channels today.** Don't try to predict the future. Instead, stay as flexible and extensible as possible.
- **Content is only as good as the creator, so consider the authoring experience.** When focusing on technical strategies for exposing content to a myriad of devices, it is easy to forget about creators. Work with creators and authors to understand how to make their experience great.
- **Define your one voice and one message.** Cross-channel marketing experiences require your brand to have a cohesive, unified, and consistent voice. Look upon this journey in content consolidation and multichannel delivery as an opportunity to affect organizational alignment and centralize content creation.
- **One size doesn't fit all.** You will have different experiences requiring various authoring approaches, delivered to an assortment of channels. The hybrid paradigm is about using the right option for a given use case. Embrace the options.

For more information:

www.adobe.com/go/aem

Appendix A: Code example showing HTTP request and JSON response.

```
HTTP GET /content/api/articles.model.json
{
    :type: “my-app/components/structure/api-page”
    title: “Data API”,
    lastModifiedDate: 1467202845038,
    templateName: “api-page”,
    language: “en-US”,
    :items: {
        root: {
            :items: {
                contentfragment: {
                    :type: “my-app/components/content/content-fragment”,
                    title: “My Article”,
                    model: “my-app/models/article”,
                    elements: {
                        title: {
                            value: “My Article”,
                            dataType: “string”,
                            title: “Article Title”
                        },
                        body: {
                            value: “... the article text... “,
                            dataType: “string”,
                            title: “Article Text”
                   },
                   ... other Content Fragment fields ...
                    }
                },
                ... Other components that expose content to this API ...
            },
            :itemsOrder: [
                “title”,
                “body”
            }
        }
    },
    :itemsOrder: [
        “root”,
    }
}
```

Appendix B. Code example showing custom component development.

```
import com.adobe.cq.export.json.ComponentExporter;

@SlingModel(
    adaptables = SlingHttpServletRequest.class,
    adapters = { MyComponent.class, ComponentExporter.class }
)
@Exporter(
    name = ExporterConstants.SLING_MODEL_EXPORTER_NAME,
    extensions = ExporterConstants.SLING_MODEL_EXTENSION
)
@JsonSerialize(as = MyComponent.class)
public class MyComponentImpl implements ComponentExporter {

    String getHelloWorld() {
        return “Hello World”;
    }

    @Override
    String    getExportedType() {
        return “my-app/components/content/my-component;
    }
}
```