A comprehensive Q&A report.

Optimizely CMS 13

April 1, 2026

Contents

Introduction

Welcome to the CMS 13 comprehensive Q&A report. This resource has been developed following the CMS 13 Technical Webinar held on March 26, 2026, to provide a single, organized point of truth for our customers and partners.

The content within this guide consolidates and deduplicates questions gathered from multiple sources. To ensure maximum utility, we have lightly edited the questions for clarity and consistent tone while strictly preserving the original technical intent.

The Q&A is categorized into seven critical focus areas to assist your planning and implementation:

- **General Product Overview**
  High-level value propositions, GA dates (March 31, 2026), and core capabilities.
- **Architecture & .NET Support**
  Deep dives into .NET 10 requirements and framework support lifecycles.
- **Optimizely Graph & Search**
  Critical guidance on the mandatory migration from Search & Navigation to Graph.
- **Visual Builder & Content Modeling**
  Insights into the new unified editing experience and "Experiences" architecture.
- **Opti ID & Authentication**
  Requirements for accessing AI capabilities and managing SSO/Identity providers.
- **Embedded DAM**
  Information on the unified asset management system and migration paths.
- **Opal & Agents**
  Details on AI-driven workflows, data privacy, and governance.

General Product Overview

What is CMS 13 and what are its key capabilities?

CMS 13 is the next major version of Optimizely's content management system, designed to help marketing and digital teams create, manage, and optimize content using AI-driven workflows and modern content architecture.

Key capabilities include Opal agents for content creation, optimization, and governance; Optimizely Graph for unified content retrieval and semantic search; Visual Builder for page creation and preview; Embedded DAM for asset management; and External Content to connect third-party systems.

What is the value proposition of CMS 13?

CMS 13 helps teams move faster with AI-assisted workflows and built-in content creation tools while maintaining quality and governance. It enables organizations to build on any frontend architecture, produce content faster with AI agents, create and personalize pages without developer dependency, connect content and data across systems, and improve discoverability across search and AI-driven channels.

Is CMS 13 a PaaS or SaaS product? How does it differ from CMS SaaS?

CMS 13 (PaaS) offers maximum flexibility for organizations that require deep architectural control and custom backend logic. In contrast, CMS SaaS is a fully managed offering where Optimizely handles the underlying infrastructure, allowing your team to focus entirely on content and delivery without managing deployments.

When will CMS 13 be available?

CMS 13 was officially released for General Availability on March 31, 2026, and is now available for all eligible customers..

Which customers are the best fit for CMS 13?

CMS 13 is ideal for enterprise organizations looking to scale their global digital presence. It’s a perfect fit for teams aiming to bridge the gap between AI-driven content velocity and enterprise-grade governance across multiple channels.

What does CMS 13 not replace?

CMS 13 is designed to sit at the center of your digital stack. It is built to integrate with, rather than replace, your preferred specialized tools for analytics, experimentation, and deep SEO auditing. You can continue using your best-of-breed third-party platforms via our robust API and connector framework.

What are the major selling points for upgrading from CMS 12 to CMS 13, particularly for clients not using Opal or Graph?

For organizations not yet leveraging Opal or Optimizely Graph, the move to CMS 13 remains a critical step for future-proofing and editorial efficiency. While Graph unlocks the full power of the new architecture, the upgrade delivers immediate value through:

- **Modern Editing with Visual Builder**: A unified, drag-and-drop editing experience that replaces traditional on-page editing and works seamlessly with existing server-side MVC rendering.
- **Long-Term Support (LTS)**: The transition to .NET 10 ensures your platform stays on the latest Microsoft-supported framework, providing better performance and security.
- **Editorial Quality-of-Life**: New tools including content templates, blueprints, and auto-translation with structure preservation allow teams to launch campaigns faster.
- **Developer Efficiency**: Includes a full REST API by default and a simplified "Applications" concept for cleaner, more scalable multi-site management.
- **Enhanced Security**: Hardened protocols for user management and file uploads to protect your digital perimeter.

Note: While CMS 13 provides significant immediate benefits, adopting Optimizely Graph (now included in your license) is recommended to eventually unlock advanced features like the new Content Manager and C# SDK.

Is there a roadmap for post-launch features?

Yes. Optimizely is committed to a high-velocity innovation cycle for CMS 13. Following the initial launch, we plan to introduce regular updates focused on enhancing Opal agents, expanding Visual Builder capabilities, and deepening integrations across the Optimizely One ecosystem. For the most current roadmap details and upcoming feature priorities, please visit: https://www.optimizely.com/product-updates/content-management/

When is CMS 14 coming and what will be in it?

With the launch of CMS 13, we are moving toward an annual major release cadence for our PaaS customers. This strategy ensures the platform remains aligned with the latest .NET LTS (Long Term Support) releases. While specific features for CMS 14 have not yet been announced, our future focus includes advancing native webhooks for better headless orchestration and continued expansion of our agentic AI framework.

Architecture & .NET Support

What .NET version does CMS 13 require?

CMS 13 requires .NET 10.

What is the .NET support strategy for CMS 12, especially given that .NET 8 reaches end of support in November 2026? Will CMS 12 officially support .NET 10?

To align with Microsoft's release lifecycle, CMS 13 is built on .NET 10 (LTS). As .NET 8 reaches its end-of-support in November 2026, transitioning to CMS 13 provides the most secure and performance-optimized path for organizations staying on a supported framework. There are currently no plans to backport .NET 10 support to CMS 12.

Are there plans for .NET 11 support for CMS 13?

.NET 11 is expected to be released in November 2026 and is a Standard Term Support (STS) release. No specific announcement has been made regarding CMS 13 and .NET 11 support.

Will Dojo widgets continue to work without changes in CMS 13?

CMS 13 maintains full backward compatibility for existing Dojo property editors. Additionally, we have modernized the extensibility model to support **ES6 modules**. This allows developers to build custom editors using modern tools like **React, TypeScript, and Vite** without any dependency on the Dojo framework. See: https://world.optimizely.com/blogs/grzegorz-wiechec/dates/2026/3/custom-property-editors-in-optimizely-cms-13/

Are Razor pages and Blazor supported in CMS 13?

While CMS 13 does not currently offer specialized 'out-of-the-box' library support for Blazor, it remains compatible with standard .NET 10 Blazor implementations. We continue to evaluate first-class support for Razor components based on community feedback.

Are there plans to support Razor components (.razor) with tag helper equivalents?

Our engineering priority for CMS 13 is the delivery of a decoupled, high-performance architecture via Optimizely Graph. While we aren't introducing new TagHelper equivalents for Razor components at this time, we remain fully committed to supporting the traditional MVC and Razor frameworks for all existing and new implementations.

Does the standard MVC implementation pattern remain supported in CMS 13?

Yes. MVC remains fully supported in CMS 13. There are no plans to drop MVC or Razor support. Tag helpers and HTML helpers are available for rendering Experiences and DAM integrations.

Optimizely Graph and Search & Navigation

Is Optimizely Graph required for CMS 13?

Yes. Optimizely Graph is a core architectural component of CMS 13. While the CMS can technically run without it, many key features, including the new Content Manager, Content Variations, and the C# SDK, require Graph to function.

Is switching from Search & Navigation to Graph mandatory when upgrading to CMS 13?

Yes. Search & Navigation is not supported in CMS 13. All search and content retrieval functionality must be migrated to Optimizely Graph as part of the upgrade.

Is there a formal end-of-life date for Search & Navigation?

No formal end-of-life date has been announced for the service itself. However, because it is not supported in CMS 13, customers migrating to the new version typically utilize a 6-month overlap period to transition their logic to Graph, with extensions available upon request.

Does switching to Graph require rebuilding the frontend?

No. CMS 13 continues to support .NET MVC, headless, and hybrid architecture. While you will need to update your data fetching logic to use the Graph API or the new .NET SDK, your existing frontend presentation layer is preserved.

What is involved in migrating from Search & Navigation to Graph?

Migration is streamlined by a new .NET SDK designed to align closely with the existing Find API. This allows for a smoother transition of search logic. A full content sync can be triggered during deployment, and a comprehensive migration guide is available to assist your technical team. See the early access preview docs: https://docs.developers.optimizely.com/content-management-system/v13-Pre-Release/docs/migrate-from-search-navigation-early-access-preview

Do you recommend moving to Graph in CMS 12 before upgrading, or waiting until CMS 13?

We recommend waiting until your CMS 13 upgrade. The Graph schema format has been optimized for CMS 13 to align with our modern SaaS standards, so migrating directly to the CMS 13 version of Graph avoids redundant development effort.

What is the upgrade effort if a site already uses Graph in CMS 12?

The effort involves adjusting existing queries to align with the new schema, where built-in system metadata is now clearly separated from implementation-defined data. Since the CMS 13 implementation is no longer based on the Content Delivery API, any custom server-side extensions will need to be reviewed and updated.

How does the Graph schema in CMS 13 align with CMS SaaS? Are there differences?

The schemas are now closely aligned. By separating system metadata from user-defined properties, both Optimizely and the implementers can evolve schemas independently without risk of breaking changes. The REST API format and payloads are consistent across both PaaS (CMS 13) and SaaS.

What is the status of Search & Navigation features (synonyms, best bets, autocomplete, spellcheck) in Graph?

The Graph Search UI includes synonym management, "Best Bet" configurations, and core analytics. Additional advanced features are scheduled for release throughout Q2 2026. Please refer to the migration documentation for current workarounds regarding tracking and spellcheck.

Will Graph support all CMS features like localization, translation, and personalization?

Yes. Optimizely Graph is designed to be a unified reflection of your content model and fully supports core platform capabilities such as localization and translation. Because Graph mirrors the CMS structure, localized content is indexed and remains retrievable based on the language settings defined within the system.

What happened to FilterForVisitor() in CMS 13?

FilterForVisitor() remains available and functions as expected in CMS 13 with no changes to its behavior. However, for teams looking to modernize their code, we recommend moving toward IContentAccessEvaluator.HasAccess(), which provides the same functionality through a more modern, dependency-injection-friendly interface.

Now that FilteredItems is gone for ContentArea, what is the best migration approach from CMS 12 to CMS 13?

In CMS 13, this property is obsolete to ensure a cleaner data model. The migration path depends on your usage:

- **Standard Rendering**: If you use HTML or Tag Helpers, no action is needed; filtering is handled automatically during rendering.

- **Custom Code**: If you called FilteredItems in code (e.g., to count visible items), you should now inject IEnumerable<IContentAreaItemsRenderingFilter> and apply the filters to the items collection directly.

What is the strategy for the Optimizely Graph .NET client package? The Optimizely.Graph.Client is deprecated, and there are new packages like Optimizely.Graph.Cms and Optimizely.Graph.Core. Can these be used for external data sources or .NET APIs hosted separately?

The new packages (Optimizely.Graph.Cms and Optimizely.Graph.Query) provide a more flexible developer experience. They are specifically designed to support broader integrations, including indexing external data sources and separate .NET APIs within the DXP environment.

How can external data be easily indexed in Graph, and is it covered under the standard license?

External data can be indexed using the modern Optimizely.Graph.Core and Optimizely.Graph.Cms .NET packages. Optimizely Graph is included as a core capability for CMS 13 DXP customers. We recommend consulting your account manager to confirm specific volume and usage limits for third-party data under your current agreement.

What is the best practice for enriching schema and data pushed into Graph?

CMS 13 utilizes Indexing Conventions and "Contracts" (abstract content types) to automatically index data. Upcoming minor releases will include further extension capabilities, allowing you to include computed properties on existing types and perform data conversions for custom property types.

Will Graph schema generation be possible locally from code before deploying or syncing content type changes?

Yes. The CMS handles schema generation from Content Types automatically. While the core system schemas are protected to ensure platform stability, extension points exist for custom properties and computed values.

Will developer Graph indexes be available for local development (similar to Search & Navigation developer indexes)?

Yes. Developer indexes are available to support your local workflows, and a self-service management portal is currently under consideration for future release.

Will there be a Graph-compatible service running as a Docker container for local development or on-prem hosted solutions?

This is currently being investigated as part of our long-term developer experience roadmap to provide more flexibility for local and on-premise environments.

Will GraphQL search include document indexing for PDFs?

Yes. Text is extracted from most common document types, including PDFs, and made searchable within the Graph index.

Will Graph leverage Virtual Roles the same way the Content Delivery API does?

Virtual roles marked as “Available as permission role” are included in Graph. Note that evaluation depends on context; some criteria require the process to be evaluated within the CMS environment rather than an external frontend.

Visual Builder and Content Modeling

Is Visual Builder available for PaaS (CMS 13) or only SaaS?

Visual Builder is a core capability of the Optimizely platform and is available for both CMS SaaS and CMS 13 (PaaS).

Can existing CMS 12 pages and blocks be used in Visual Builder?

Yes. Visual Builder supports existing pages and blocks from CMS 12. While shared blocks currently function as standalone assets rather than nested elements within sections (support for nested shared blocks is on the roadmap), your existing content library remains fully functional. We recommend reviewing your content modeling during the upgrade to maximize the layout flexibility offered by Visual Builder.

Can the CMS 12 page/block architecture be combined with Visual Builder Experiences in CMS 13?

Yes. CMS 13 supports a hybrid approach, allowing you to utilize traditional Pages and Blocks alongside the new "Experiences" and "Sections" architecture. Static properties can also be added to Experiences for additional metadata management.

What is the row/column structure in Visual Builder, and is it mandatory?

Rows and columns provide a logical grouping within the Visual Builder interface (Sections > Rows > Columns > Elements). This structure is designed for editorial ease and does not dictate how the content must be rendered on your frontend; your development team maintains full control over the final presentation.

Do Elements within Columns support ContentArea, or only ContentReference?

Elements inside Columns utilize ContentReference rather than ContentArea. This alignment ensures a more performant and structured content model, consistent with the architecture used across the Optimizely platform.

Is On-Page Editing (OPE) supported in CMS 13?

Yes. On-Page Editing remains available as a configurable option in CMS 13. Comprehensive documentation and best practices will be provided at launch to help you configure the editing experience for your teams.

How does On-Page Editing work in CMS 13 for decoupled/headless implementations?

For decoupled or headless environments, the editing experience is primarily powered by Visual Builder. It features a split-view interface where the left pane serves as the editorial control center and the right pane provides a high-fidelity live preview. This includes support for granular componentization and the ability for editors to view changes across different device breakpoints.

Key technical aspects include:

- **Granular Componentization**: Visual Builder allows for a cleaner content model by letting developers componentize content at a more granular level than traditional blocks.
- **Preview Breakpoints**: Just like in CMS 12, editors can view their changes across different device breakpoints within the live preview.
- **Support for Existing Structures**: Visual Builder supports pages and blocks from CMS 12, meaning no immediate changes are required for existing content structures during migration.
- **Content Modeling Agent**: An AI-driven agent is available to recommend and create content models optimized specifically for Visual Builder based on design files, URLs, or prompts.
- **Graph Integration**: The editing experience is increasingly dependent on Optimizely Graph, which serves as the delivery and integration layer for features like the embedded DAM and external content.

How much of experience pages can be read or updated using IContentLoader/IContentRepository?

Both IContentLoader and IContentRepository fully support "Experiences," allowing your developers to programmatically manage these new content types just as they do with traditional pages and blocks.

How do Visitor Groups and Audiences work in CMS 13?

Visitor Groups and Audiences continue to work as expected in MVC-based implementations. For delivery via Optimizely Graph, as they are not indexed, we recommend utilizing the new Content Variations and Optimizely Experimentation to deliver high-performance personalized experiences.

Will Experimentation and Personalization be more closely embedded into the CMS platform in CMS 13?

Yes. In CMS 13, AI-driven capabilities through Opal are more deeply embedded into the core user experience. This is supported by Optimizely Graph, which enables advanced personalization and semantic search by allowing the system to query content dynamically based on the specific user context.

Opti ID and Authentication

Is Opti ID required for CMS 13?

Opti ID is required to unlock centralized platform features, including the Opal suite and the Embedded DAM. It is the standard identity protocol for Optimizely's SaaS and DXP (PaaS) environments.

Is Opti ID mandatory for CMS 13 editors in PaaS?

Yes, Opti ID is the required authentication method for business users (editors and administrators) to access CMS-level features such as the Asset Library and AI agents. For a detailed breakdown of specific editor workflows, please refer to the CMS 13 launch documentation.

What are the actual hard requirements for Opti ID? Can it be configured alongside traditional login methods?

Opti ID is mandatory for internal business users accessing the CMS management interface within the DXP. However, for your public-facing site visitors or end-users, you may continue to use any preferred authentication scheme (such as Azure AD B2C or custom providers). Note that Opti ID is not available for on-premise or self-hosted Azure installations.

We use Okta across our company for SSO, including the CMS. How would we manage that with Opti ID?

Opti ID is designed to integrate with your existing enterprise identity strategy. You can configure Okta as an external Identity Provider (IdP) for Opti ID. This ensures that your staff can continue to use their existing corporate credentials to access the Optimizely platform seamlessly.

We plan to allow customers to authenticate via our own B2C system for seamless passthrough to non-Optimizely systems. How does this interact with Opti ID requirements?

These two systems operate independently. Opti ID manages access for your "business users" (the people building the site), while your B2C system manages access for your "site visitors" (your customers). CMS 13 fully supports this side-by-side configuration.

How will Opti ID work with multi-site setups in local development? The limited callback URLs in CMS 12 significantly limit local multi-site testing.

We are aware of the requirements for local multi-site development. We are planning a self-service portal where development teams can manage redirect URIs and callback URLs more flexibly to improve the local testing experience.

Will Opti ID offer better integration with Microsoft Entra ID in the future?

Opti ID already supports integration with Microsoft Entra ID (formerly Azure AD) as an Identity Provider. If your organization has specific functional requirements or identified gaps, we encourage you to submit that feedback through your Customer Success Manager.

Does CMS 13 enforce idle session timeouts and prevent multiple concurrent sessions (issues observed in CMS 12)?

Yes. Opti ID provides centralized control over session behavior. Administrators can adjust configurations to manage idle timeouts and session concurrency to align with your organization’s security policies.

Embedded DAM

Is the Embedded DAM available for PaaS (DXP) customers?

Yes. The Embedded DAM is available for all CMS 13 DXP customers and is accessed through the Opti ID authentication service.

Can the DAM be activated for any DXP customer without additional cost?

Yes. The Embedded DAM is included as a core capability within the standard CMS 13 and CMS SaaS license packages.

Is the DAM a cloud-only service, or can it be hosted on-prem?

The Embedded DAM is a cloud-native service. Because it requires Opti ID for authentication and management, it is not available for on-premise or self-hosted installations.

Does the Embedded DAM replace the modal popup present in the CMS 12 DAM integration?

The new Content Manager picker is designed to eventually succeed the legacy modal interface. In the initial release of CMS 13, the existing library picker remains available as an option to ensure a smooth transition for editorial teams.

When using DAM, can the old media library be disabled?

In the initial release of CMS 13, both the local CMS media library and the Embedded DAM can coexist. This allows teams to transition assets at their own pace. A toggle to fully disable the legacy media library is planned for a future update.

Are there migration scripts or tooling to migrate blob storage assets from CMS to the DAM?

There is no "one-click" native migration tool, as asset migration involves updating references throughout your content. However, because both libraries can coexist, a manual or phased migration is fully supported. For large-scale automated migrations, specialized third-party tools (such as the Proxima DAM Migration Tool) are available through the Optimizely partner ecosystem.

Will the DAM support consuming images through Cloudflare image resizing functionality?

The DAM product team is actively exploring advanced image processing and delivery optimizations as part of the ongoing product roadmap.

Will there be support to restrict DAM/CMP assets by user role or Visitor Groups?

In the initial release of CMS 13, external content (including DAM assets) is globally accessible within the CMS. Granular role-based access control for specific DAM assets is a prioritized item on the post-launch roadmap.

Will DAM allow custom properties or additional metadata on assets?

Yes. In CMS 13, the DAM integration is built on Optimizely Graph. This means all tags, labels, and custom fields defined within the DAM are indexed and made available as metadata that can be queried and utilized directly in your content delivery.

Opal and Agents

Will Opal work with PaaS (CMS 13)?

Yes. Opal is available in CMS 13 (PaaS) with the same range of out-of-the-box agents and tools currently available in the SaaS version.

Is Opal available for on-prem customers?

No. Opal requires Opti ID for authentication and secure connectivity, which is not supported for on-premise installations.

Will it be possible to publish content directly from Opal in CMS 13?

Yes. Opal agents can connect directly to the CMS API, enabling them to create, edit, and publish content based on your specific prompts and workflows.

Is there an additional cost for Opal in CMS 13, and do all CMS 13 customers get it automatically?

Opal is an optional, opt-in feature. It requires a specific provisioning process and the installation of the supporting integration packages. Usage is managed through an Opal credit system; please contact your Customer Success Manager (CSM) to discuss the credit package that best fits your organization's needs.

Can an AI chatbot be built using Opal Agents that answers user questions based on website content?

For Editorial Use (Internal): Yes. OpalChat is available to help editors interact with and query their own content within the CMS interface. See: https://support.optimizely.com/hc/en-us/articles/37791100847373-2026-Optimizely-Opal-release-notes#h_01KMZY72QQJRWK7432XE5GEPQB

For Website Visitors (External): Opal is currently focused on empowering the editorial and marketing workflow. Using Opal as a public-facing frontend chatbot for end customers is not a supported use case at this time.

How does Opal handle multi-brand governance within a shared agent workflow?

Opal agents are highly configurable. By utilizing a combination of specific instructions and prompt engineering, you can ensure that agents adhere to unique brand guidelines and governance standards across different business units.

Is the AI/agentic direction in CMS 13 baked into the platform or an optional add-on layer?

Opal is an optional, high-value layer. It must be explicitly provisioned and installed to appear within your CMS environment, giving you full control over when and how your organization adopts AI.

Will Opal have data residency options, and can clients opt out of Opal?

Opal is strictly opt-in; it will only appear if your organization chooses to provision it. Once active, administrators can disable Opal or restrict access to specific users at any time. Regarding data residency, services are currently US-based, with EU-based residency options arriving shortly.

If sensitive data is involved, are there guarantees that Opal will not store or train on that data?

Yes. Any data your organization inputs into Opal remains private to your organization. Our underlying LLM provider, Google Cloud, does not utilize customer inputs or LLM outputs for its own model training purposes. Furthermore, your data, including both Opal inputs and outputs, is not used by Optimizely to train, develop, or improve any general generative AI features. All processing occurs within Google Cloud's secure, enterprise-grade infrastructure to ensure the highest level of data protection.

Will there be an MCP server for CMS, Graph, and CMP for integration into developer tools like Cursor or Claude Code?

Yes. A release of a CMS MCP (Model Context Protocol) server is on the roadmap. Because the MCP is an open standard, this server can be utilized by any compatible LLM or client, including modern developer tools like Cursor and Claude Code. This integration will allow developers to surface CMS context, schemas, and content directly within their AI-assisted coding environments, significantly streamlining the development and integration lifecycle.

Migration and Upgrade Paths

What is the recommended upgrade path from CMS 12 to CMS 13?

The transition involves four primary strategic steps: (1) Configuring Opti ID for centralized identity, (2) Enabling Optimizely Graph to power content delivery, (3) Updating content models for those utilizing Visual Builder, and (4) Migrating assets for teams adopting the Embedded DAM. While timing varies based on implementation complexity, typical enterprise upgrades range from 2 to 6 months. Detailed technical documentation is available to guide your team through each phase. See documentation for more information: https://docs.developers.optimizely.com/content-management-system/v13.0.0-CMS/docs/upgrade-to-cms-13

What are the most common breaking changes when upgrading to CMS 13, and what remediation steps does Optimizely recommend?

As with any major version release, breaking changes are focused on modernization. The most significant shifts involve the move to .NET 10 and the replacement of legacy search libraries with Optimizely Graph. We recommend reviewing the Breaking Changes Documentation early in your planning cycle to identify specific code patterns, such as legacy content area filtering, that require remediation. See documentation for more information: https://docs.developers.optimizely.com/content-management-system/v13.0.0-CMS/docs/breaking-changes-in-cms-13

Which authentication and integration components require mandatory updates or replacement when moving to CMS 13? Is Opti ID mandatory?

Opti ID is required to enable Opal capabilities and the Embedded DAM. Additionally, Search & Navigation must be replaced by Optimizely Graph, as legacy search providers are not supported in CMS 13. A full list of component dependencies is provided in our migration guide.

Are there recommended tools or automated approaches for refactoring deprecated APIs and legacy initialization/configuration code?

We are currently exploring automated migration tooling and AI-assisted refactoring options to streamline the upgrade process.

Is a direct upgrade from CMS 11 to CMS 13 supported, or is going through CMS 12 recommended?

A direct upgrade from CMS 11 to CMS 13 is technically supported. Because the majority of the effort involves the transition between .NET framework versions, the level of work is similar to an 11-to-12 upgrade. However, for highly customized solutions, we recommend consulting with your implementation partner to determine if an intermediate step through CMS 12 is advisable for risk mitigation.

Is a project migration process available in the PaaS Portal (similar to the CMS 11-to-12 migration)?

Yes. Comprehensive documentation for the project migration workflow within the PaaS Portal is published alongside the CMS 13 release to assist with environment provisioning and data transfer.

Can a CMS 12 instance run in parallel with a CMS 13 instance during migration?

Yes. We support parallel environments to allow for rigorous testing and validation. This is specifically recommended when transitioning to Optimizely Graph to ensure search parity before the final production cutover.

What changes can be made in CMS 12 now to prepare for a future CMS 13 upgrade?

The most impactful step you can take today is adopting Opti ID for your business users while still on CMS 12. This streamlines your authentication early and ensures your team is ready for the platform's new centralized identity model.

Regarding the search and delivery strategy, we recommend waiting until your CMS 13 upgrade to transition to Optimizely Graph.

For agencies currently delivering CMS 12 projects, what is the realistic minimum-effort path to CMS 13, and are there any gotchas beyond the documented breaking changes?

The minimum-effort path focuses on framework compatibility and search migration. We are actively monitoring early upgrades and continuously updating our public documentation to address common edge cases. The full list of breaking changes is the most critical resource for identifying potential "gotchas" in custom codebases.

When will CMS 12 reach end of life (EOL)?

Optimizely does not formally "end-of-life" major versions; however, we focus our primary support and new feature development on the current major version. We backport critical security fixes to the previous version (CMS 12) for a specified window, but we strongly encourage customers to stay current on CMS 13 to benefit from the latest performance, security, and AI updates.

Will the Episerver Foundation template (https://github.com/episerver/Foundation) be updated for CMS 13?

Yes. Foundation remains our premier reference implementation. Our Solution Architecture team is currently updating the repository to CMS 13 to demonstrate best practices for the new architecture, including Visual Builder and Graph integrations.

What is the upgrade path for clients who are still on-prem? Will there be official documentation on which CMS 13 features are unavailable for on-prem customers?

The upgrade path for on-premise customers follows the standard .NET 10 transition. However, customers should be aware that cloud-native features, specifically Opti ID, Opal, and Optimizely Graph, require a DXP environment. Our documentation clearly outlines these dependencies to help on-premise customers plan their architecture accordingly.

What is the release cadence between CMS 13 PaaS and SaaS going forward?

We are moving toward an annual release cadence for PaaS. This ensures that our PaaS customers receive major platform innovations and framework updates (like .NET LTS releases) more frequently, keeping the PaaS and SaaS offerings closely aligned.

Commerce Connect 15

Is Commerce 14 compatible with CMS 13?

No. Commerce 14 is not compatible with the architecture of CMS 13. To upgrade to CMS 13, you must also upgrade to Commerce 15.

When will Commerce 15 be available?

Commerce 15 is officially targeted for general availability in late April 2026. For teams planning their migrations now, a preview version is currently available via our developer documentation to assist with early integration testing. https://docs.developers.optimizely.com/commerce-connect/v15-pre-release/docs/overview-of-commerce-15-pre-release#/versions

Is there a recommended upgrade order for Commerce and CMS?

We recommend upgrading to Commerce 15 either prior to or in conjunction with your CMS 13 upgrade. This ensures that the shared dependencies between the two products remain synchronized throughout the transition to .NET 10.

Will the Commerce Service API continue to be supported in Commerce 15?

Yes. The Commerce Service API remains a supported component. A Commerce 15-compatible version of the Service API is scheduled for release in Q2 2026.

Are there performance improvements for large product catalogs in Commerce 15? Some sites experience very slow catalog updates with millions of products and attributes.

Commerce 15 is primarily a compatibility release designed to align with CMS 13 and .NET 10. However, it does include targeted performance improvements for catalog language handling. These optimizations can provide significant benefits for enterprise organizations managing high-volume, multi-language product catalogs.

Will Experimentation and Personalization be embedded more closely into Commerce 15?

In Commerce 15, the core experimentation and personalization workflows remain centralized within the CMS. This allows marketing teams to utilize Optimizely's full experimentation suite on commerce-related content and product data as it is surfaced through the CMS and Optimizely Graph.

Add-ons, Packages, and Third-Party Compatibility

Which packages officially supported by Optimizely will be available and compatible with CMS 13 at launch?

Most core Optimizely-supported packages are being updated for CMS 13 compatibility. While many are available at launch, others, such as Forms (Classic), are scheduled for release shortly after the initial CMS 13 rollout. We are also working closely with our partner ecosystem to ensure that popular partner-led packages are updated in tandem with the new release.

Do you plan to upgrade add-on packages such as Find, Forms, and LanguageManager to support CMS 13?

Forms (Classic): Will be supported via an update shortly after the CMS 13 launch.

LanguageManager: An updated version is scheduled for public release by the end of Q2 2026.

Search & Navigation (Find): There are no plans to update the Find package for CMS 13. The transition to Optimizely Graph is the required path for search and content retrieval in the new version.

Is there a plan to maintain technical support for the Find package in CMS 13 to allow a phased migration away from Search & Navigation?

No. The Search & Navigation (Find) packages will not be upgraded to support CMS 13. While Search & Navigation remains fully supported for sites on CMS 12, all CMS 13 implementations must utilize Optimizely Graph. We recommend completing your search migration as part of the upgrade to CMS 13.

Will Dojo widgets continue to work without changes in CMS 13?

Yes. Existing Dojo property editors continue to work in CMS 13 to ensure backward compatibility. Additionally, CMS 13 introduces a modern, ES6-based approach for custom property editors, allowing developers to build new extensions without any Dojo dependencies.

Will TinyMCE be updated in CMS 13?

Yes. CMS 13 includes an update to TinyMCE 8, bringing the latest rich-text editing features and security improvements to the platform.

Will third-party add-ons such as Geta categories, NotFoundHandler, Geta Generic Links, and Advanced Preview be compatible with CMS 13?

Compatibility for third-party and community add-ons is managed by their respective owners. We recommend checking the GitHub repositories for these packages for the most current status. For example, the author of Advanced Reviews has confirmed that CMS 13 support is forthcoming. For community-led Geta packages, we encourage submitting inquiries via their respective repository "Issues" pages.

Will security patches continue for supported Optimizely packages (e.g., NotFoundHandler) while customers are waiting to upgrade to CMS 13?

For packages provided officially by Optimizely, we follow the same lifecycle policy as the CMS major version. We will continue to provide critical security patches for CMS 12-compatible versions of our add-ons while CMS 12 remains a supported version. Please note that community packages (such as NotFoundHandler) are maintained by third parties and do not fall under Optimizely's official support policy.

Will Product and Content Recommendations features work in CMS 13?

Product Recommendations are fully expected to be supported in CMS 13. Content Recommendations utilize a lightweight integration that is anticipated to remain compatible; however, we recommend verifying specific implementation details with your account team during the upgrade planning phase.

Developer Experience and Local Development

Are there plans to improve the CMS experience on mobile devices? Many buttons are currently unusable on mobile.

While the current focus for CMS 13 is on the foundational architecture and the new Visual Builder desktop experience, we recognize the importance of mobile usability. There are no specific mobile-first feature announcements for the initial release, but the move to modern UI frameworks in Visual Builder provides a better foundation for future responsive improvements to the editorial interface.

How should Graph be configured in a development environment? Will a Graph instance be required per developer?

To support local development, Optimizely provides a self-service portal where teams can request short-lived developer indexes. These allow individual developers to index and query their local content changes without impacting the shared staging or production Graph instances. See https://docs.developers.optimizely.com/content-management-system/v13.0.0-CMS/docs/cms-13-overview?isFramePreview=true#request-a-developer-instance

Will developer Graph instances be included in DXP the way Search & Navigation developer indexes were?

Yes. We are automating this process to match the developer experience provided by Search & Navigation. By the end of Q2 2026, a dedicated portal will be available for developers to request and provision demo/dev instances instantly via a self-service form. In the interim, Optimizely Support can assist with manual provisioning of these instances.

Are the new HTML/tag helpers documented anywhere?

Documentation for the new Tag Helpers, specifically those used for Visual Builder Experiences and the Embedded DAM, is being finalized.

Will there be a React skeleton project for Optimizely that supports inline editing in Visual Builder?

Shortly after the initial release, we will launch an updated version of the Content JS SDK specifically designed for CMS 13. This SDK will include samples and templates for headless React projects. Please note that while headless setups are fully supported in Visual Builder, the "inline" editing experience differs from traditional MVC; headless implementations utilize a real-time preview with floating property editors for a streamlined editorial flow.

Will there be a CMS 13 template similar to Alloy for a quick-start empty project?

Yes. We are introducing a new starter template called Stride. Stride is built specifically for CMS 13 and is fully optimized for the new Visual Builder content model. The classic Alloy template has also been updated to support the move to .NET 10.

Are there guides or resources for switching from a Razor frontend to a headless React/Next.js setup?

Yes. We recommend starting with our updated JS SDK documentation. It is important to note that moving from a Razor-based (MVC) architecture to a headless (Next.js/React) setup is a significant architectural shift that involves rebuilding the frontend layer. Our guides focus on how to map your existing content types to the new Graph-powered delivery API to simplify the data transition.

Is it possible to generate the Graph schema locally from code before deploying in a PaaS-based setup?

In CMS 13, the schema is generated automatically by the CMS based on your Content Types. While you cannot currently "pre-generate" the schema locally without an active Graph instance, the system is designed to keep your code and schema in sync. An Indexing Conventions API is scheduled for a minor post-launch release, which will allow for more granular control over how computed properties and extensions are reflected in the schema.

On-Premises Considerations

Is Opti ID available for on-prem environments?

No. Opti ID is a cloud-native identity service and is only available for customers in DXP (SaaS or PaaS) environments.

Can on-prem customers upgrade to CMS 13, and what features will be unavailable to them?

Yes, on-premise customers can upgrade to CMS 13 to take advantage of the .NET 10 framework and core CMS improvements. However, because Opti ID is a prerequisite for several platform-level features, on-premise installations will not have access to Opal, the Embedded DAM, or the Optimizely Connect Platform. While Optimizely Graph is required and runs in the cloud, it can still be accessed and utilized by an on-premise CMS instance.

Will on-prem customers be able to purchase a standalone Graph instance to use CMS 13 Graph features?

Optimizely Graph is a managed cloud-hosted service and is not available for self-hosting or on-premises deployment. However, it is designed for connectivity; any CMS 13 instance with an internet connection can connect to and utilize the Graph service, regardless of whether the CMS itself is hosted on-premise or in a private cloud.

Can a CMS 13 instance be run locally for development purposes?

Yes. CMS 13 (PaaS) projects can be fully configured to run on local development machines. It is important to note that connectivity to the internet is required to utilize cloud-dependent features like Opti ID and Optimizely Graph during the development process.

APIs, SDKs, and Technical Architecture

Are there any changes to the IContentEvents event system in CMS 13?

No, the IContentEvents system remains unchanged in CMS 13. However, looking ahead, we anticipate shifts in this area for CMS 14, driven by the ongoing development of native webhooks to better support decoupled architectures.

What is the replacement for SiteDefinitions (IApplicationResolver)? Is there an equivalent of ISiteDefinitionRepository for the new Application model?

Yes. In CMS 13, the legacy site definition services have been modernized. IApplicationResolver and IApplicationRepository now serve as the primary services for resolving and managing the new Application model.

PageReference is being deprecated in CMS 13. What should be noted when transitioning to ContentReference?

When transitioning to ContentReference, you can achieve the same restrictive behavior as PageReference by using the AllowedTypes attribute. Simply set the allowed types to PageData to ensure users can only select pages within that property.

Are there changes to how custom Admin pages are created, managed, or rendered in CMS 13?

No. The creation and management of custom Admin pages remain consistent with the patterns established in CMS 12, ensuring a smooth transition for your existing administrative extensions.

Will scheduled jobs be made async in CMS 13?

The Scheduled Jobs API has not changed for the 13.0 release. While we have plans to provide full support for asynchronous scheduled jobs in a future update, the current implementation remains synchronous.

Is the SaaS CMS REST API available for PaaS (CMS 13), and are there any differences?

Yes. The REST API utilized in our SaaS offering is available in CMS 13 with the exact same format and payload structure, providing a unified developer experience across both deployment models.

What is the status of Content Delivery API support in CMS 13?

The Content Delivery API is supported for backward compatibility but will not be included in the initial CMS 13.0 release; it is expected in a subsequent minor update. Optimizely Graph remains our recommended long-term delivery approach.

What is the Content JS SDK strategy for CMS 13? Is it the preferred way to manage CMS models, and does it support headless PaaS scenarios?

A new version of the Content JS SDK is being released to support both SaaS and CMS 13, and it is the recommended approach for frontend model management. For PaaS-based headless scenarios, we recommend managing types where they are primarily utilized. While full PaaS headless support is an area of ongoing development, the JS SDK provides a robust bridge for most modern frontend requirements.

Where should the source of truth for CMS content models live in a headless PaaS setup using the JS SDK?

In a PaaS + Headless configuration, the recommended source of truth is typically your .NET code. The JS SDK is then used to synchronize those types to your frontend, ensuring your content models remain consistent across the stack.

What is with PropertyList<T> and PropertyDefinitionTypePlugIn in ContentGraph? They are not included in the schema when T is based on a POCO class.

In ContentGraph, custom properties default to their underlying base types; for PropertyList<T>, the schema type is indexed as JSON. It is worth noting that PropertyList<T> has effectively been superseded by IList<TBlock>, which provides a significantly better interface and more robust indexing behavior within Graph.

Is ConventionRepository the best way to get StandardContentBase classes into ContentGraph?

To ensure base types are correctly reflected in Graph, we recommend modeling them as Contracts (also known as abstract content types). Content types modeled as contracts are automatically indexed in Graph, providing a cleaner and more reliable integration than using ConventionRepository.

What are the best practices for UI extension and extending the CMS 13 UI?

CMS 13 maintains support for the UI extension patterns used in CMS 12. However, we have introduced a powerful new method for creating custom property editors that removes the need for Dojo. Developers can now build rich UI extensions using modern JavaScript (ES6), React, or TypeScript. https://world.optimizely.com/blogs/grzegorz-wiechec/dates/2026/3/custom-property-editors-in-optimizely-cms-13/.

How will CDA-based Graph customizations (previously used for backwards compatibility with Search & Navigation) be achievable in CMS 13 + Graph?

The initial release of CMS 13 does not support direct schema extensions or computed properties within the Graph index. However, a new Indexing Conventions API is scheduled for a minor post-launch release, which will enable the capability to extend the schema and add computed properties.

Headless and Decoupled Implementations

How does the preview pane work in Visual Builder for headless implementations?

Visual Builder supports live preview for headless setups by communicating with your external frontend via a secure postMessage interface. Detailed technical documentation on enabling and configuring live preview for your headless environment is available on our documentation.
https://docs.developers.optimizely.com/content-management-system/v13.0.0-CMS/docs/enable-live-preview

Will CMS 13 have better integration for frontend frameworks like Next.js?

Yes. The updated Content JS SDK is specifically designed to provide a more seamless integration experience for Next.js and other modern JavaScript-based frontends. It simplifies data fetching from Optimizely Graph and helps manage content structures within your JS application.

What is the recommended way to handle content types when running PaaS + headless with the new JS Content SDK?

The general rule is to manage types where they are primarily used. For a standard .NET PaaS implementation, content types should continue to be managed in C# code. The JS SDK is then used to synchronize these types to your frontend via the Optimizely Graph schema, ensuring your data models remain consistent across the stack.

Is Blazor supported for headless CMS implementations in CMS 13?

While there is no official SDK or dedicated "first-class" support library for Blazor at this time, its use is not discouraged. Developers can successfully implement Blazor-based frontends by utilizing the Optimizely Graph .NET SDK or the native .NET APIs for content retrieval.

Will there be a React skeleton project for Optimizely CMS 13 that supports inline editing in Visual Builder?

Shortly after the 13.0.0 release, we will launch an updated version of the JS SDK that includes samples and templates for React. It is important to note that "inline" editing (direct text manipulation on the page) is not supported for headless setups. However, headless is fully supported in Visual Builder, where editors use a real-time preview combined with floating property editors to manage content.