CMS MACH CAPABILITY SCORE

Instructions: Complete the set of 30 questions on the next 2 pages of this document, get your scores, and see where your CMS candidates fall on the MACH maturity scale.

Results by Category

Microservices	
API-First	
Cloud-Native	
Headless	

Average

MACH MATURITY SCALE

3

Leader

The CMS is natively designed with MACH principles. Supports a composable enterprise in which every component is pluggable, scalable, replaceable, and can be continuously improved through agile development to meet evolving business requirements. Highly adaptable to integrations and content delivery channels.

2.5

Learner

The CMS does not have a natively MACH foundation. The vendor is aware of the competitive benefits of MACH design, and is beginning to add a MACH layer to the CMS to meet modern needs. Allows for integration with other tools and modern content delivery channels, but there are limitations due to parts of the underlying architecture not being accessible via APIs.

2

Laggard

The CMS has a monolith design that has APIs added for select functionalities. Implementation, integrations, maintenance, and upgrades will likely require substantial effort, cost, and specialized developer talent. Suited mainly for traditional, page-based website delivery.

1.5

Left Behind

The CMS is most likely a legacy technology that is part of a wider suite of solutions. Implementation and maintenance of the CMS may require a dedicated team. Integrations with customer experience technologies outside of the vendor's suite will be difficult.

1

For each question, fill in your score here ▼

MICROSERVICES

Question	*Score*	*Score*	*Guidelines*
Are you able to use functionality as discrete services?		3	Yes, you can use functionalities and capabilities independently.
		2	Some functionalities can be provided as services, but others require to have the whole platform in place before using a particular service.
		1	No, functionalities are baked into the platform and not available as services. You must implement the entire solution to access any functionality.
Are the services decoupled?		3	Yes, the services do not depend on one another. If there is an issue with one service the overall solution remains structurally stable.
		2	Services may be in separate application environments, such as content management and content delivery, but they share common dependencies.
		1	All functionality is dependent on each other.
Can services be scaled independently?		3	Yes, services can be scaled on a granular level without impacting infrastructure. There is no need for application level scaling.
		2	Services can not be scaled independently, but it may be possible to scale sections or tiers of the application. It is possible to scale the platform at an application level which might require additional resources.
		1	No, if more resources are needed the whole application, and the hardware it uses, has to be multiplied to meet demand.
Are services and functionalities upgraded independently?		3	Yes, services are upgraded and deployed with no impact to your implementation. Upgrades can happen daily with zero downtime.
		2	Releases and upgrades come in bundles. You are expected to test the compatibility of an updated service with your solution.
		1	Upgrades happen all at once when a new version of the CMS is released. The cost and effort of upgrading lies with the customer.
Does the platform support a modular implementation?		3	Yes, you can launch a project using a selection of services and gradually add or replace additional services over time. Supports an iterative approach.
		2	You can implement in stages but you need to create a blueprint of how the full solution will fit into your ecosystem before getting started. If you want the opportunities to add features or functionality in the future, you must plan for these up front.
		1	Implementation commonly requires a full rip and replace of your previous CMS.
Are you able to add services from third party tools into your solution?		3	Yes, microservices can be added, replaced, or removed as needed with no change to the back end.
		2	You can use services from other tools if defined up front. Switching functionalities down the line requires back-end changes and might result in limited support of the core platform.
		1	You can add third party functionality through heavy customization.
Are you able to interchange any service, such as search or check out, without impacting other functionalities?		3	Yes, you are free to substitute any component with a third party option. Services can be interchanged with no impact on the platform.
		2	You can use services from other tools if defined up front. Switching services down the line might require breaking changes.
		1	No, the functionality in the platform is specifically designed to work together, a substitution would greatly decrease performance.
Are you able to create a composable solution ecosystem?		3	Yes, the CMS focuses on content management and is designed to integrate with services from other best-of-breed tools.
		2	The CMS is able to integrate with other tools but it is highly suggested that you use the platform, and the native features, as your all-in-one customer experience tool.
		1	The CMS is one part of a suite of tools that combine to cover the entire digital experience.
Microservices Average			

API-FIRST

Question	*Score*	*Score*	*Guidelines*
Is 100% of functionality in the CMS available through a set of APIs?		3	Yes, it is possible to control the entire platform through APIs without requiring a UI.
		2	There is a content delivery API, but some features in the UI and back-end are not available via an API.
		1	There is an API for certain content functions, but the software is generally black box.
Can you implement a solution with any technology stack / programming language / front-end framework?		3	Yes, language and toolset agnostic. There are multiple SDKs available to make working with technologies easier, but you are not limited to just the SDKs available
		2	It is possible, but it is also required to use a particular stack language for certain functionality.
		1	The solution must use specific and possibly proprietary technologies and templating languages.
Can the developer do any coding without calling APIs?		3	No, all functionality happens via APIs.
		2	Yes, there is custom platform code alongside APIs.
		1	The majority of development happens with code custom to the platform.
Can all functionality be extended or integrated through standard APIs or events?		3	Yes, fully supported. Web APIs are decoupled and standardized.
		2	Some integrations happen via APIs, others integrations require a plug-in and/or customization.
		1	No, integrations require software plugins or custom development.
Are all new features directly released as APIs?		3	Yes, features are immediately available through APIs.
		2	Features are released in the platform first and an API may be offered later.
		1	New features require an upgrade of the platform.
Are APIs backwards compatible?		3	Yes, all APIs have versioning and there are no breaking changes.
		2	There is a single API that is tied to the platform version.
		1	APIs change between upgrades and may require extensive upgrade effort.
How much time do you have to adopt a change in API version?		3	The APIs are fully versioned, giving you time to migrate.
		2	The lack of versioning requires that you adopt breaking changes immediately.
		1	You adopt new APIs when you upgrade to a full new version of the platform.
Are all APIs fully documented?		3	Yes, documentation is clean, clearly shows capabilities, and is easy to develop with.
		2	There is limited documentation, especially outside of the core API.
		1	API Documentation is an afterthought.
Are GraphQL endpoints supported the same as traditional REST APIs?		3	Yes, there is GraphQL coverage.
		-	-
		1	No.
API-First Average			

CLOUD-NATIVE

Question	*Score*	*Score Guidelines*	
What is the delivery model of the solution?		3	Multi-tenant Software-as-Service (SaaS).
		2	Managed Hosted.
		1	On-Premise.
Do you have to provision environments for new customers? If so, how long does that take per environment?		3	There are no dedicated environments, new environments are provisioned instantly.
		2	A new environment has to be created per customer.
		1	You are responsible for your own environment.
Must extensions/customization to the platform be deployed to a specific cloud vendor/location?		3	For your custom solution, you can deploy to the cloud of your choice.
		2	You must deploy your solution on a cloud infrastructure selected by the vendor.
		1	You are responsible for your own environment.
Are you involved in upgrading the environment (platform / infrastructure)?		3	Upgrades happen continuously on both functionality and infrastructure. Upgrades are available automatically without manual effort and there is no downtime for upgrades.
		2	Infrastructure upgrades are automatically provided, but the platform might require a manual effort to install or upgrade. There is downtime when upgrades occur.
		1	Upgrades are infrequent and are not always backwards compatible. Upgrading the the CMS may require development effort to make the upgrade compatible with your wider solution.
What is the estimated cost for the effort of an upgrade project?		3	You receive upgrades automatically, there is no cost.
		2	There is a cost associated with upgrades.
		1	Upgrading the platform requires a large amount of development effort and potentially additional license overhead.
How often are new features and upgrades released?		3	There is proof (blogs, documentation, webinars, etc) of a steady stream of feature releases.
		2	Features are released in bundles 1-4 times a year.
		1	Features are released only when there is an new version of the entire product.
When upgrading, do you have to coordinate testing with the vendor?		3	No.
		2	Yes, you must coordinate testing to ensure there are no breaking changes to the wider solution.
		1	Upgrading versions is usually facilitated by a full project.
Is it possible to install the product on-premise?		3	No, the product has always been exclusively a cloud offering
		2	Yes, the original version of the product was an on-premise version.
		1	A majority of the customer base uses the on-premise solution.
Cloud-Native Average			

HEADLESS

Question	*Score*	*Score Guidelines*	
What type of APIs support headless content consumption?		3	Content and all management functions can be consumed, created, and modified via a full featured set of native, high performing APIs.
		2	There is a REST API to consume Content-as-a-Service but its querying options are limited. There is no write or management option.
		1	Content can be exported via batch tools but not accessed real-time via APIs.
Is there an option of front-end delivery within the platform?		3	No. All front-end delivery requires a third party.
		2	Yes, the original front-end delivery tool has been “decoupled” with APIs and you can use either method. Sometimes called a “hybrid” option.
		1	Yes, and it is strongly suggested that you use the front-end delivery in the platform.
Can you choose any front-end framework?		3	Yes, you can work with any framework. There are software development kits (SDKs) and guides available for multiple front-end tools, but you can work with frameworks even if there is no SDK yet.
		2	You can work with any framework, but the vendor might have built a limited number of integrations with front-end frameworks.
		1	There may be one or two front end tools that have integrations built, but it is highly suggested to use the vendors delivery tier.
Does the platform support additional delivery channels?		3	There are SDKs for a variety of front-end providers, mobile apps, and traditional application programming languages. APIs are channel agnostic.
		2	There are a variety of SDKs available. The APIs are not fully channel agnostic, and adapting to newer channels (i.e. voice, AR) would require customization.
		1	New channels require extensive customization or waiting for the vendor to adopt them into the platform.
Do you need vendor specific code to make a front end work with the CMS back end?		3	No, everything is delivered through standardized APIs that are agnostic to third parties.
		2	Yes, front-end developers need to know specific back-end knowledge about the vendor’s code to make it work.
		1	Using a third party for front-end delivery would require customization on the back end.
Headless Average			