commercetools
Next generation commerce

Migration from Oracle Commerce to commercetools

www.commercetools.com

Table of Contents

Table of Contents

Executive Summary

It's past time to move off of Oracle Commerce (formerly known as ATG). For a host of reasons, which we'll discuss shortly, Oracle Commerce is the wrong stack for this generation of commerce. This whitepaper leads architects and their teams through the platform migration from Oracle Commerce to commercetools.

Introduction

In today's fast-moving commerce business, brands and merchants need to enhance their agility and flexibility. Customer demand is continually changing, new touchpoints are emerging, and innovative ways of communication between consumers and suppliers emerge. Take voice commerce enabled by devices with voice assistants. Within only a few years, more than 200 million smart speakers have been sold. Similar to the App Store which helped Apple shape a new age of app-driven mobile devices, Amazon lets third-party developers build new capabilities on top of their voice platform. Brands and retailers need to be able to build new prototypes quickly, experiment with the user experience, and create great services for their customers and drive loyalty.

In reality, however, many organizations are kept from building new business models by their slow, hard-to-adapt software platforms which effectively stifle innovation. The only way out is to move to another, more flexible architecture.

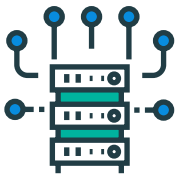
In this whitepaper, we look specifically at two platform vendors on different ends of the spectrum: on the one hand, there is the Oracle Commerce suite, which was recognized as one of the market leaders by Forrester and Gartner but has now reached its end of life, hinted by Oracle's continuous downsizing of the team working on its Commerce Cloud suite through multiple layoffs. On the other hand, there is the innovative cloud-native commercetools platform, which has shot up the ranks among analysts and reviewers, and is recognized as a leader by both Forrester and Gartner in the commerce software space. We will suggest a strategy on how to move from Oracle Commerce to commercetools without disrupting daily business or jeopardizing operations.

Of course, there is no standard way of migration and we're by no means suggesting a cookie-cutter approach since every project is different and has its caveats. Rather, the following strategy is meant as a guiding principle or a framework which, from practical experience, tremendously increases the chances for success. These are the six steps we will be covering in this guide:

Step 1 — *Discovery and Gap Analysis*

Step 2 — *Build Migration Roadmap*

Step 3 — *Data Modeling*



Step 4 — *Build Custom Extensions*

Step 5 — *System Integration*

Step 6 — *Integrate the User Experience*

A New Approach to Commerce

In order to understand the differences between the commercetools and Oracle Commerce platforms, there are two concepts you should be familiar with. These are what we believe to be key differences between the two platforms and come into play as we discuss moving your business to commercetools.

The MACH Standard

Modern commerce platforms are built differently than their counterparts that were built over 20 years ago. commercetools is a founding member of the MACH Alliance, which means it adheres to 4 guiding principles as a platform. These principles are core to enabling the platform to be performant, flexible and easy to work with.

The MACH Four

Technologies that are disrupting the e-commerce software market.

M

Microservices

A modern architecture that makes your IT team agile, possibly even SUPERSONIC.

A

API-First

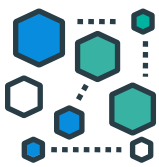
100% API-centric means you can incorporate any functionality.

C

Cloud-Native

Huge promotions, giant traffic spikes - no worries. Our multi-tenant, cloud-native platform is always there.

H



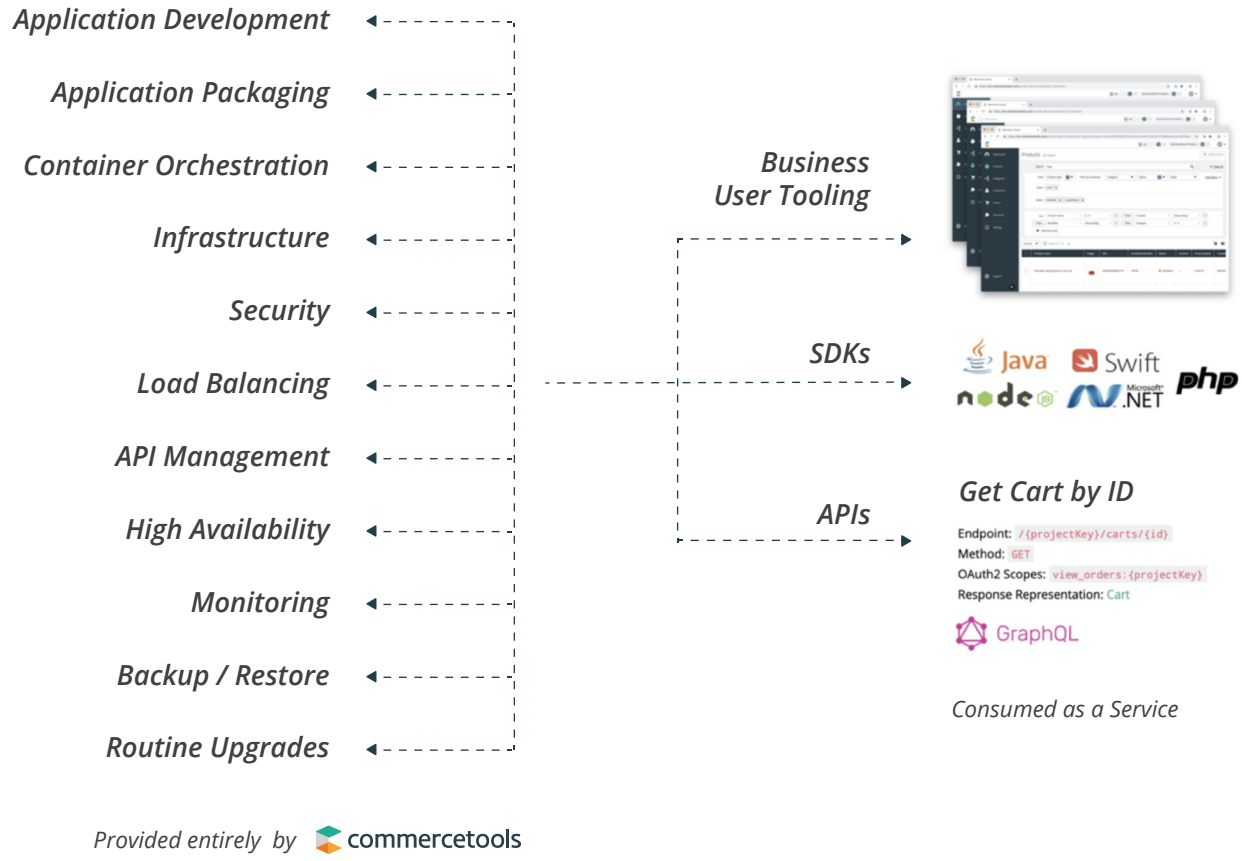
Headless

Choose your own frontend: build from scratch or buy a packaged DXP. Put your customer experience front and center - we've got both ends covered!.

Enterprise SaaS

Along with platforms following the MACH standard, a new approach to how your platform is implemented and managed is taking hold in the market. In the previous generation, platforms were traditionally installed “on-premise”, or in other words, hosted and managed by you. This required large IT teams to take care of things like hardware provisioning, databases, network security, monitoring, patching, upgrading, and so forth. The checklist you need to manage to host a commerce platform is extensive and comes with a constant expense of having an extensive team with the right people. Primarily such expense is just to make sure it’s a secure and performant environment.

Today, enterprise SaaS (Software-as-a-Service) solutions are mainstream. You no longer need to have an operations team to handle “keeping the lights on”. At commercetools, we have a world-class team managing the multi-tenant platform for your specific implementation and the team has a proven track record of success. So much so that we contractually guarantee our platform will be up and performant for every customer.



Reasons for Wanting to Migrate

Business Agility

Because of the complex nature and the strict architecture of the Oracle Commerce platform, developers need to work with many different layers. For example, a rather trivial task such as building a custom promotion and displaying it in the frontend often takes backend developers a few days to implement. This is mainly due to a lot of boilerplate code and the fact that the ORM data modeling is done via XML configuration.

	Oracle Commerce	*commercetools*
Customization	Experienced Oracle Commerce developers needed to customize	Developers work with well-documented APIs
Technology	Java, JSP, XML configuration	Language-agnostic, SDKs available for Node, Java, PHP, .NET and others

Total Cost of Ownership (TCO)

Typically, Oracle Commerce is sold in the form of core-based licenses. Even for mid-sized companies, seven-figure licensing fees are not unheard of. For brands and retailers, this is a challenge, especially when it comes to scaling: if more hardware (i.e. more cores) is needed because the business is growing and the traffic is peaking, more licenses have to be purchased.

Businesses also need to consider that, as the Oracle Commerce core platform evolves, there are compulsory updates which need to be performed as well. Oracle would release new major versions of the platform on a yearly basis and these updates were not trivial to apply. The customizations you have made to the platform have a high likelihood of not being compatible with the new version, hence requiring code changes and re-testing of existing features. This is a major pain and often requires six-figure investment. Nevertheless, these upgrades are necessary to be and remain eligible for support – which, in the form of a support contract, also generates recurring costs.

Last but not least, the budget for running and maintaining an Oracle Commerce installation has to be kept in mind as well. The platform needs to be run on multiple application nodes and database servers, generating operating costs for hosting applications.

	Oracle Commerce	*commercetools*
Licensing	Previously core-based	Usage-based
Operations	Multiple application and database nodes necessary to run at scale	Platform run and scaled by commercetools; only frontend hosting as an external cost
Updates & Upgrades	Compulsory updates a few times a year	Continuous integration, features pushed daily, non-breaking API changes

Platform Performance / Scalability

The architecture of Oracle Commerce makes it difficult for the solution to meet the performance and scalability requirements that most mid-market and enterprise businesses have. Like most monolithic applications, Oracle Commerce is deployed using a three tier architecture: web servers, application servers and a database. This architecture makes it difficult to scale because when you add more web servers, you might not have enough capacity with your application servers and if you add more application servers, you might not have enough database capacity. Add that together with the complex web server load balancing configurations, the database connection limits in your application server configuration, the transaction timeouts, etc. and you can quickly see why it takes most companies running Oracle Commerce approximately 2-3 months to prepare for the peak season traffic events. This takes a big chunk of time and money away from new initiatives that will actually generate more revenue.

	Oracle Commerce	*commercetools*
Deployment Architecture	Multi-tier, high level of dependency between tiers	Microservices, individually deployable and scaled
Runtime Environment	Threaded application, synchronous execution	Asynchronous processing, event-driven communication
Read/Write Operations	Large distributed transactions in the application, database locking at the row level	Separation between read and write stores (CQRS), eventing sourcing

Choosing the Right Frontend

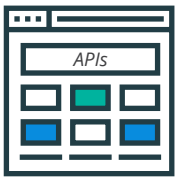
A headless frontend gives you the confidence to move quickly and the freedom to shape your brand. By having the frontend decoupled from your commercetools headless commerce platform, your team can be removed from the limitations of a monolithic stack and be free to change and innovate the frontend at an accelerated pace.

A headless frontend delivers faster, more engaging digital experiences with a best-of-breed approach that improves conversion and customer loyalty. You won't be limited to a frontend structure prescribed by the commerce platform so you have the freedom to customize the experience to shape your brand's identity. Meanwhile, the pace of frontend innovation can move faster, letting you quickly adopt the latest customer experience technology and rapidly react to market trends or shopper expectations.

There are many options for choosing a frontend, at a high level, they fall into three broad categories, as shown in this diagram:

Suite Approach

DXP

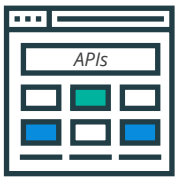


Headless Commerce Platform

- ✓ | *Time-to-Market: 3 Months*
- ✓ | *Technical Skills Required: Low*
- ✓ | *Robust features (Personalization, CMS and Advanced Merch)*
- ✓ | *Non-MACH Stack*

FaaS

Frontend-as-a-Service

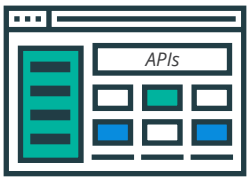


Headless Commerce Platform

- ✓ | *Time-to-Market: 4-6 Months*
- ✓ | *Technical Skills Required: Medium*
- ✓ | *Basic CMS Capabilities*
- ✓ | *MACH Stack*

Custom Front-End

Custom Frontend



Headless CMS | *Headless Commerce Platform*

- ✓ | *Time-to-Market: 6-9 Months*
- ✓ | *Technical Skills Required: High*
- ✓ | *Robust CMS Capabilities*
- ✓ | *MACH Stack*

Suite Approach

The suite approach utilizes a DXP (Digital Experience Platform) that provides an all-in-one solution for delivering experiences to your customers. DXP suites often contain a combination of features and business user tools, such as drag-and-drop layout management, personalization, search & merchandising tools, etc. The suite approach works well for companies with small IT teams or who are looking for an “out of the box” solution - it may not work as well for companies that have a large team of frontend developers who want to build their own experiences.

Frontend as a Service

The FaaS (Frontend as a Service) approach utilizes solutions which provide the frontend (customer-facing) experience without all the bells and whistles of a full DXP, but with capabilities such as a business user interface for managing the layout, look and feel, etc. This is a good solution for companies with a moderate level of frontend development capabilities that also want an ‘out-of-the-box’ solution for business users to manage layout and content.

Custom Frontend

One of the biggest benefits of the commercetools platform is that 100% of the functionality is exposed via APIs. Effectively giving frontend developers the flexibility and freedom to develop any type of user interfaces they need for all of the different touchpoints/channels in which they wish to interact with their customers.

In the case of web and mobile web channels, for example, there has been a proliferation in the emergence of frameworks built on top of the popular node.js architecture. Examples include Angular, React, Vue.js, etc. commercetools provides an open-source Single Page Application built on Vue.js for developers looking for a reference application to help them get started:

https://github.com/commercetools/sunrise-spa

We also provide an iOS reference application for iOS developers looking to build native iOS apps:

https://github.com/commercetools/commercetools-sunrise-ios

To speed development and simplify development, we also provide client SDKs in popular languages such as Java, JavaScript, PHP, iOS, and .NET:

https://docs.commercetools.com/software-development-kits

The Frontend Vendor Landscape

When you choose the commercetools headless commerce platform, you have the ultimate freedom and flexibility to build out your frontend. Using any of the approaches described above or a combination of approaches, depending on how you wish to address all of the different touchpoints where your customers interact with your brand. For an up-to-date list of all of the frontend and CMS vendors with productized integrations to commercetools, take a look at our Integration Marketplace:

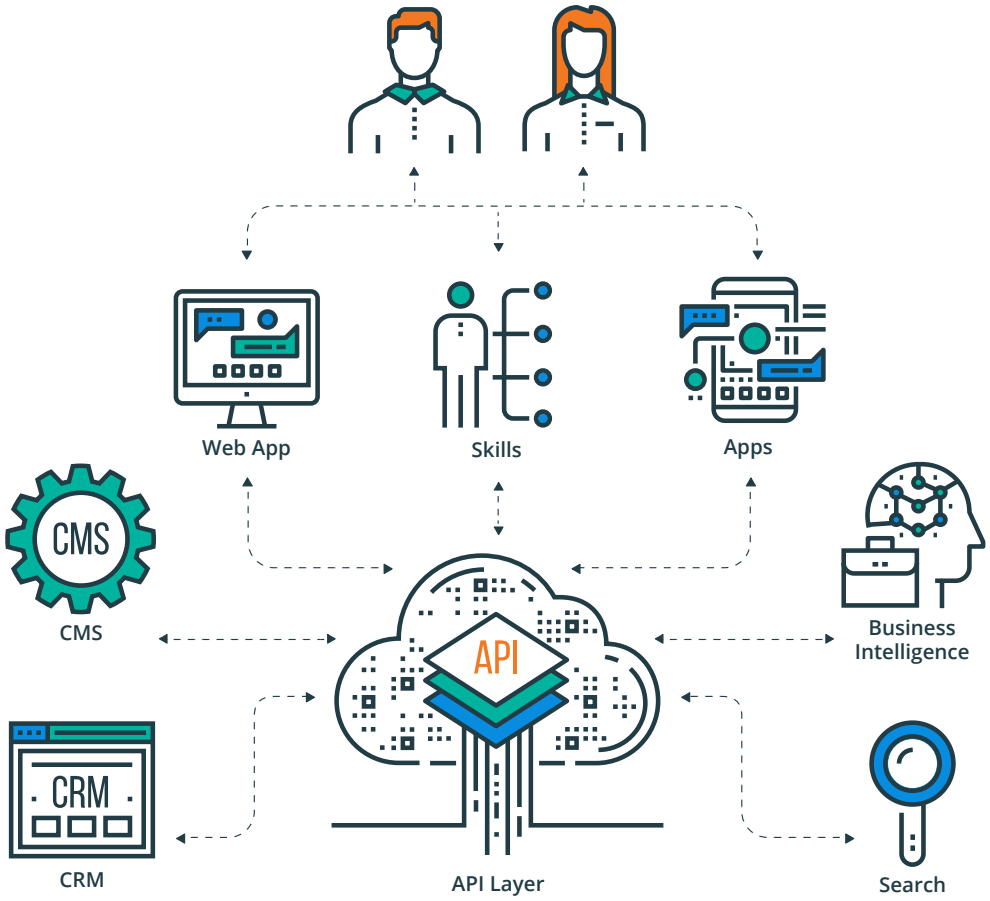
https://marketplace.commercetools.com/integrations/frontend

Migration Planning and Execution

There is no set-in-stone definition of what a platform migration means in the commerce space. For most people, the ideal outcome would be a “like-for-like” solution; meaning everything previously possible in the old solution and every feature that has been developed, would be there in the new solution as well – plus the benefits of the new platform. In most circumstances, as we will see a little later on, this expectation is not practical: when planning the migration, this is the perfect chance to eliminate overhead and generally aim for a leaner solution.

What we’re suggesting here is to divide an existing project into business domains and to transfer the respective functionality and data out of ATG to a best-of-breed infrastructure with commercetools at its core.

In most cases, this means: Going from on-premise monolith to a service-oriented, headless cloud solution, with everything which is connected to this project. We are also aiming at a phased migration (instead of a big bang approach) to minimize disruptions to operations and mitigate risks to the best extent possible.



Step 1: Discovery and Gap Analysis

Now getting into the first step of the migration, discovery, and gap analysis. This is a crucial step to ensure you've determined the full scope and put a project plan together. Here's a list of objectives for this step:

1. Taking stock: What does the current platform offer, which kinds of functionalities does it enable, which user stories does it support, which processes are running in the background? These questions might sound obvious or trivial, but with a code-base which is a few years old, there are always individual edge cases that might not have been documented properly or which people have just forgotten about. Also, it is highly recommended not to shift your entire digital business all at once: focus on less important assets first, like a less relevant locale, before tackling the core business.

2. Setting priorities: Create a list of all of those processes and edge cases, before deciding which of those have to be migrated immediately and which ones can be tackled at a later stage – or even dropped altogether. Make sure to view these items from a business perspective and determine which value they're adding to your business. Even if you are tempted to stick with this feature that took ages to implement and that everybody on your team is proud of, if it does not add value it probably shouldn't be on the list.

3. Gap analysis: Use this prioritized list of functions and processes to hold it against the features which are present out-of-the-box in commercetools. You will find detailed information in the API documentation (https://docs.commercetools.com/http-api.html). There are mainly three options for each item:

- **Out-of-the-box:** The desired feature is a standard commercetools feature that can be configured.
- **3rd party:** There are features and domains which are out of the commercetools scope, such as CMS/DXP. In these cases, 3rd party services need to be integrated.
- **Customized:** These are the parts that have to be custom-coded.

4. Organize teams: Finally, make sure your staff and your organizational structure meet the new demands of a cloud-based solution. Instead of clear-cut, horizontally organized skill sets – such as frontend specialists, backend developers, data scientists – the "new cloud world" requires the work of cross-functional and vertical teams.

Step 2: Build a Migration Roadmap

The first thing to do is build a migration roadmap that lists important milestones, deliverables, and a timeline. Roughly speaking there are three areas which are the basis for the roadmap:

1. Data: One of the first things you should tackle is making sure to move data from the Oracle Commerce database to commercetools, namely the following:

- **Product catalog** - catalogs, categories, products, SKUs, etc
- **Customer profile** - customer segments, customer profile, address, payment methods, etc
- **Orders** -carts, orders, shipping methods, etc.

We will talk more about what to do in these cases in Step 3 of the migration plan.

2. Business logic: This area includes all the custom extensions which need to be built or integrated when 3rd party services are included.

3. UI/UX: commercetools is a headless commerce platform. This means that there is no fully-fledged storefront application like in Oracle Commerce. However, our flexible APIs help you rapidly prototype, connect the frontend(s) of your choosing and build customized customer experiences, not just for web and mobile, but for any touchpoint out there, from in-car displays to smart home appliances and voice assistants.

We also offer an Accelerator program (https://commercetools.com/accelerator) that allows you to pair our commerce platform with your choice of a frontend from several partners to get up and running quickly.

***Note:* that those three areas do not have to be tackled in this exact sequence. These suggestions only serve to add a bit of structure to your planning.**

Now with the planning done, we get to the execution tasks of the migration. At this point you should have a list of requirements to be migrated to the new platform to work off of. Multiple streams can be done in parallel to reduce the overall duration of this part.

Step 3: Data Modeling

One of the primary tasks to be completed during migration is defining your data model in the new platform. This generally doesn't map 1-to-1 and there are opportunities for optimization you should take advantage of during this step. There is a statement I would like to empathize strongly: **defining a proper data model upfront will significantly decrease the overall effort of the migration.** Why is this the case? If you cut corners and try to apply the Oracle Commerce data model, you'll end up with a mess in commercetools and that will make all of your frontend and backend integrations more difficult. I'll go into more detail in the next couple sections on how these two platforms handle data modeling differently.

How Oracle Commerce data model is defined

The data model in Oracle Commerce is defined in the Data Anywhere layer, which is a repository definition layer that uses an XML file-based format for the definitions. You have the ability to extend the OOTB Oracle Commerce objects, create entirely new objects and relationships between those objects. You also have the ability to do some tricky things like map an external interface to an object in the platform (revolutionary for its day).

If we take an example of adding of a basic change to the User Object, like adding a"Twitter Handle" attribute, this is what that change looks like in Oracle Commerce:

```
<item-descriptor name="user" xml-combine="append" item-expire-timeout="3600000" >
    <table name="user_ext" type="auxiliary" id-column-name="user_id">
        <property name="twitterHandle" display-name="Twitter Handle" column-name="TWITTER_HANDLE"
            category="BASICS" data-type="string"/>
    </table>
</item-descriptor>
```

This defines we are extending the 'user' item descriptor, we've created a new table in the database called 'user_ext' and there's a new property defined that is linked to a column in that table. There's a lot going on there. This also leads to 1000s of database tables scattered around and is a nightmare for DBAs. You don't want to fire up an instance of Oracle Commerce every time you want to access data and most of the time have to reverse engineer how to access the data if you go directly to the database.

Additionally, a change to the data model can be a time-consuming process. It requires your development team to modify the application code to add the definition, your database administrator to modify the schema in the database, and your operations team to deploy the new version of code and restart the application servers.

How commercetools data model is defined

Unlike Oracle Commerce, commercetools uses an extremely flexible data model that allows for real-time updates to its structure. Rather than changing a table in a database and then mapping it in the repository definition layer, you can change the data model in real-time using the Merchant Center or directly at the API level. There are three main concepts in commercetools to understand with data modeling:

1. Types: Allow you to create subtypes of an existing object. You assign a Type as a data record is created when you want it to be of a specific subtype. For example, you could define a t-shirt product subtype that captures the size (S/M/L/XL) and a jeans product subtype that captures inseam/waist size. When creating products, you could then create a generic product, a t-shirt product, or a jeans product. You can apply this concept to other objects, including categories, customers, orders, etc. See https://docs.commercetools.com/http-api-projects-types for more information.

2. Custom Fields: Custom fields allow you to extend existing objects when you don't want to create subtypes. You can add any supported types of fields to existing objects. See https://docs.commercetools.com/http-api-projects-custom-fields for more information.

3. Custom Objects: Custom objects in commercetools are arbitrary JSON-formatted records that are persisted indefinitely. They can be identified by a key or an ID and can be nested using the container attribute. Attributes of a custom object can reference other objects in commercetools, like an order or customer profile. See https://docs.commercetools.com/http-api-projects-types.html for more information.

Taking the Twitter example from above, this is the same extension to the User object. It creates a Custom Field on Customer.

```
{
  "key": "customer-ext",
  "name": { "en": "Extension to Customer" },
  "resourceTypeIds": ["customer"],
  "fieldDefinitions": [
    {
      "type":{
        "name":"LocalizedString"
      },
      "name":"twitterHandle",
      "label":{
        "en":"Twitter Handle"
      },
      "required":false,
      "inputHint":"SingleLine"
    }
  ]
}
```

As you can see, this data model extension uses a JSON format and it gets applied via REST API. You would use this specific API for creating the extensions:
https://docs.commercetools.com/http-api-projects-types#create-type

Adapting APIs to your data model

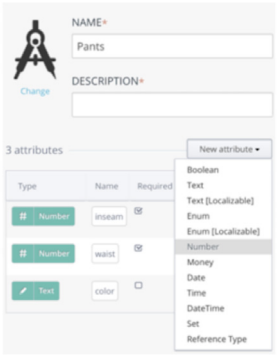
By migrating to a modern platform, you should expect there to be improved ways of doing things, right? That's exactly the case here. Whenever you modify the data model in commercetools, you don't have to take any additional steps to map that data model to the API endpoints. The platform intelligently adapts for you, saving a serious amount of time for your team. Referring back to the example of adding a "Twitter Handle" to the customer model, you can expect that field to be present in the response as soon as you've made the changes to the data model.

Step 4: Customize Platform Behavior

The next logical step is to extend the commercetools platform by implementing your business logic. This is handled differently with commercetools than it is for Oracle Commerce. As described previously, the commercetools platform is a SaaS solution and you won't customize the core code in the platform. Instead, you'll use a number of ways we provide to allow you to satisfy your business's requirements.

Here's a visual guide to the common approaches to extensions:

Extend Data Model (Fields & State)



Microservices: Inherently synchronous

Events (Subscriptions): Inherently asynchronous

Functions (API Extensions): Inherently synchronous

UI Extensions

Extending API Behavior

With an API-based platform, customizing the platform essentially means to extend the API behavior. No platform will ever be exactly what you need and this is a necessary step for every project. With commercetools, we try to make it as easy as possible, without sacrificing the ability to extend the platform when you need to. This is accomplished with two different approaches:

1. API Extensions: An API Extension allows you to inject additional logic into the server-side processing of the platform. Let's say for example you need to validate the maximum quantity of an item a customer can purchase or you need to validate inventory in an OMS before placing an order. Both of these cases are handled with API Extensions. You can find tutorials with examples of how API Extensions are implemented here: https://docs.commercetools.com/tutorial-extensions or a link to the documentation of API Extensions here: https://docs.commercetools.com/http-api-projects-api-extensions.html

2. Microservices: The other common approach is to write a microservice wrapper to sit on top of the commercetools APIs. Let's say you want to provide an API response in a very specific format, you can expose new endpoints with a wrapper. These wrappers can make a single call to commercetools or multiple calls if complex datasets are needed. Or even in other cases, you may want to combine the response of commercetools with another system. Our Blueprint Architecture provides an example of wrapper services you can use as a starting point.

Extending Business Tooling

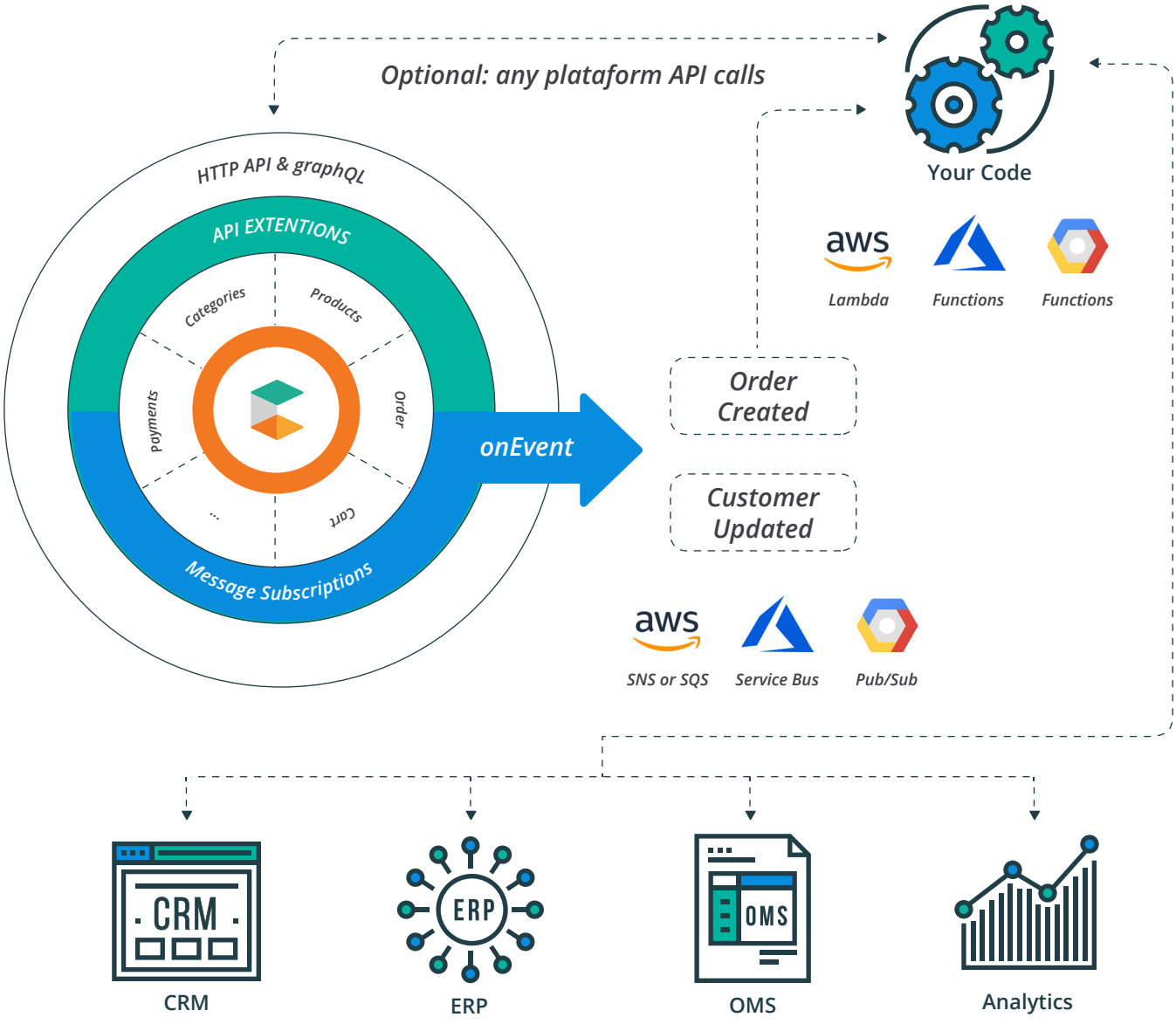
It is also common that you'll need to extend or customize the business tooling provided by the platform. commercetools comes with a BCC-equivalent called the Merchant Center. We provide the ability to extend this tool to suit your needs. This might include the ability to create new management interfaces, create specific reports important to your operations, or even exposing tooling that interacts with third-party systems to provide a single interface for your business to use.

More information can be found here: https://docs.commercetools.com/custom-applications/

Step 5: System Integration

Nearly all of our customers have at least one system they need to integrate with on the backend. This may be receiving product data from a PIM or sending orders to an OMS. With Oracle Commerce, there is not an easy way to support integrations and it was usually left to the developers to build from scratch. Techniques in Oracle Commerce range from building a separate integration module or even trying to go directly to the database.

Integrations with commercetools become much more lightweight and loosely coupled thanks to our eventing architecture. Our Subscription feature allows you to “subscribe” to events within the platform that may need to be communicated to other systems.



More information on Subscriptions can be found here:
https://docs.commercetools.com/tutorial-subscriptions

Step 6: Integrate the User Experience Layer

Integrating the user experience layer (frontend) is generally the most time-consuming part of the migration. Depending on whether you have already separated out your frontend from Oracle Commerce or not makes a big difference here. If you have, we've seen this portion take as little as a few weeks to map it completely to commercetools. If you haven't separated it out yet, you'll most likely be looking at a complete rebuild. We do have a number of assets to help you if in this case.

commercetools Reference Store (B2C) - built as a Single-Page Application using Vue.js and GraphQL

https://github.com/commercetools/sunrise-spa#sunrise-as-a-single-page-application

Vue Storefront - open-source PWA for eCommerce with prebuilt integration with commercetools

https://www.vuestorefront.io/

We also have partnerships and pre-built integrations with most of the CMS/DXP industry-leading solutions. You can find more information on our provided integration here:
https://marketplace.commercetools.com/

Blueprint Architecture

At commercetools, we know there can be a learning curve with a new platform and introducing your team to many new concepts. Based on that, we provide a Blueprint Architecture that provides real examples of all the concepts we've discussed in this document. This is a great resource for our customers and can shave 6-8 weeks off your implementation.

You can read the Blueprint Architecture that is available here:
https://ok.commercetools.com/blueprint-architecture

Conclusion

In this whitepaper, we have suggested a framework for helping you migrate your digital assets from Oracle Commerce to commercetools. As we have mentioned, this is not a one-click-solution but a project involving many aspects. Especially when it comes to moving custom functionality and individual user interfaces where most artifacts will have to be built from scratch. A positive point is that the migration of data – especially catalog data – can be at least semi-automated. Last but not least, this kind of solution allows you to build a highly effective and scalable network of services, enabling you to innovate and grow your business.

To stay relevant for their customers, they cannot have technical teams deal with maintenance and updates – instead, they need to build customer-facing features that generate real tangible business value.

What's Next?

60-day Trial

The commercetools trial provides risk-free evaluation with fully-functional access to our platform for 60 days at no cost. The trial includes access to our APIs, Merchant Center administration interface, and our tools/connectors for integrations. We believe in full transparency with our customers and we are the only enterprise commerce platform offering this level of self-service evaluation. Along with access to the platform, you can also access our documentation, tutorials, release notes, and platform status without the need to create an account. Get started today at commercetools.com!

About commercetools

commercetools is a next-generation software technology company that offers a true cloud commerce platform, providing the building blocks for the new digital commerce age. Our leading-edge API approach helps retailers create brand value by empowering commerce teams to design unique and engaging digital commerce experiences everywhere – today and in the future. Our agile, componentized architecture improves profitability by significantly reducing development time and resources required to migrate to modern commerce technology to meet new customer demands.

The innovative platform design enables commerce possibilities for the future by offering the option to either use the platform's entire set of features or deploy individual services, á la carte over time. This state-of-the-art architecture is the perfect starting point for customized microservices, enabling retailers to significantly reduce time-to-market for innovative commerce functionalities.

With offices in Germany and the United States, as well as presence across general Europe and Asia Pacific/Oceania, B2C and B2B companies from across the globe including well-known brands across many industries, including fashion, food and retail, trust commercetools to power their digital commerce business.

Germany commercetools GmbH
Adams-Lehmann-Str. 44
80797 Munich
Germany
Phone: +49 89 9982996-0
Email: mail@commercetools.com

United States commercetools Inc.
American Tobacco Campus | Reed Building
324 Blackwell St. Suite 120
Durham, NC 27701
Phone: +1 212-220-3809
Email: mail@commercetools.com

www.commercetools.com

Munich - Berlin - Jena - Amsterdam - London - Durham NC - Singapore - Melbourne