commercetools
*Next generation commerce*

# Moving from a monolithic commerce platform to commercetools

commercetools.com

# Executive Summary

*This white paper leads architects and their teams through the platform migration from a monolithic legacy commerce platform such as SAP Hybris or IBM Websphere to commercetools. It talks about which steps are necessary in this process as well as how to build a migration roadmap, migrate data and deal with custom extensions and user interfaces.*

# 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: within only a few years, more than 100 million smart speakers have been sold. Similar to the app store which helped Apple introduce the mobile age, Amazon lets 3rd-party developers build new capabilities on top of their voice platform. Brands and retailers need to be able to build new prototypes quickly, experiment, 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: move to another, more flexible architecture.

In this whitepaper, we will suggest a strategy on how to move from a monolithic commerce platform to commercetools without disrupting the daily business or jeopardizing operations.

Of course, there is no standard setup which could be migrated in a standard way. Every project is different and has its own caveats, and we're by no means suggesting a cookie-cutter approach. Rather, the following strategy is meant as a guiding principle or a framework which, from practical experience, tremendously increases the chance for success.

# Commerce Platform Migration

## Reasons to Migrate

There are two main reasons why businesses are deciding to move away from a monolithic platform:

### Total Cost of Ownership (TCO)

Typically, on-premise solutions are sold in the form of core-based licenses. Even for mid-sized companies, six-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 their legacy platform evolves, there are compulsory updates which need to perform well. Software vendors release new versions a few times per year, and from experience are often making fundamental changes to the core, such as changing internal APIs. In other words, businesses need to review and rewrite parts of their custom code to make sure it also works with the new version. This is a major pain and often requires six-figure investment. Nevertheless, these upgrades are necessary to be 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 a monolithic 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.



| | Monolithic platform | commercetools |
|---|---|---|
| Licensing | 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 external cost |
| Updates & Upgrades | Compulsory updates a few times a year | Continuous Integration, features pushed daily, non-breaking API changes |

### Lack of Agility and Speed

Because of the complex nature and the strict architecture of monolithic platforms, 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.

## The Right Time for a Migration Project

After all, introducing a new technology always has the potential to disrupt operations. On the other hand, such a move always enables brands and merchants to re-evaluate whether their current IT ecosystem is able to support current and future business objectives.

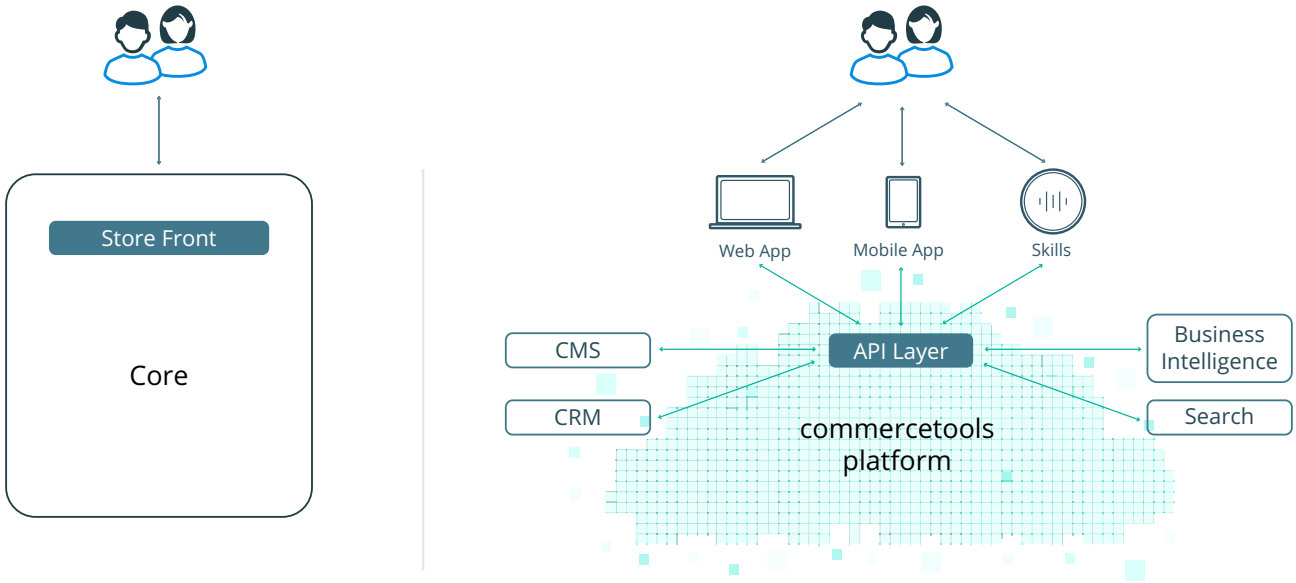
A good time for evaluating other software solutions is when businesses are facing a compulsory update. If they know that this process involves investing in new developments anyway (see the TCO section), they might as well opt for a complete change and consider moving to a technology such as commercetools.

Let's now dive deeper on how, in practice, such a migration is planned and carried out.

# 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. 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 which had not been used and generally aim for a leaner solution.

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




We are also aiming at a phased migration (instead of a big bang approach) to disturb operations as little as possible and mitigate risks.

# Step 1: Discovery and Gap Analysis

In this first step, you could structure your tasks like this:

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 which 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 and decide which of those have to be migrated immediately, and which of those 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. (And 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 and compare it against the features which are present out-of-the-box in commercetools. You find detailed information in the API documentation (https://docs.commercetools.com/http-api.html). There are 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 which have to be custom-coded.

4. **Build 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

Next, build a migration roadmap which 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 legacy database to commercetools. The three types of data include:

    - **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 although we offer a quickstart template called SUNRISE (https://commercetools.com/sunrise), but a flexible API which can be used to build customized web or mobile apps.

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



# Step 3: Extract Data

Next, diff that exported data model with commercetools' data model (https://docs.commercetools.com/http-api.html). You will want to note any differences in:

- **Objects:** a pet food company might have a "dog" object
- **Sub-types**: a clothing retailer might have shirts and jeans product types, each with different attributes
- **Object attributes:** a digital product might have a "download count" attribute

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 back, you can change the data model in real-time using the Merchant Center or directly on API level.

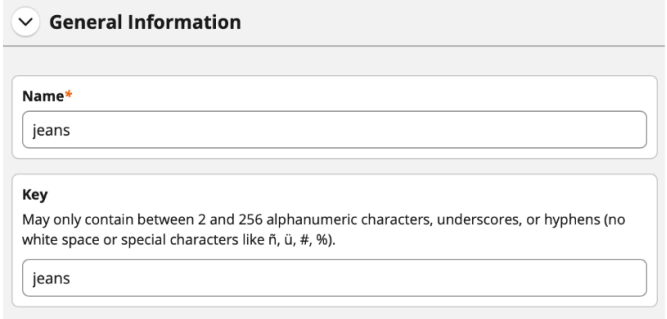
## Custom Objects

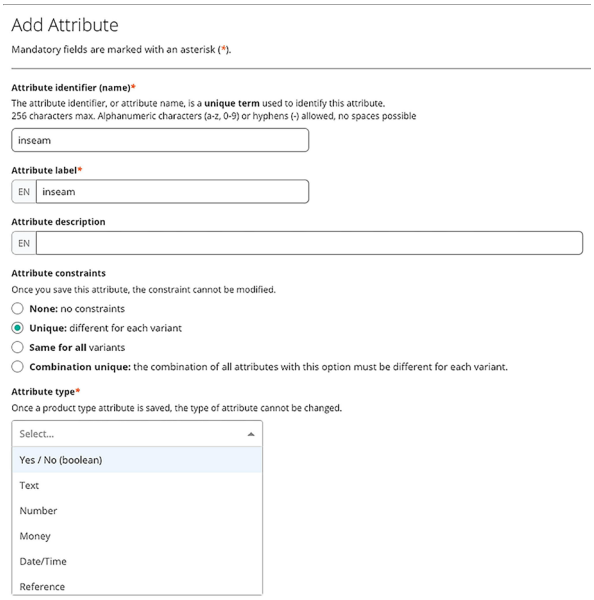
Custom objects in commercetools are arbitrary JSON-formatted records that are persisted indefinitely. They can be identified by a key or an ID and can be nested using the container attribute. Attributes of a custom object can reference other objects in commercetools, like an order or customer profile.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

## Sub-types become Custom Types

Sub-types are represented as custom types in commercetools. 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.

- Tutorial: https://docs.commercetools.com/tutorial-custom-types.html
- Documentation on types: https://docs.commercetools.com/http-api-projects-types.html
- Documentation on product types https://docs.commercetools.com/http-api-projects-productTypes.html




Finally, you can model attributes using “custom fields” (https://docs.commercetools.com/http-api-projects-custom-fields.html) in commercetools when you don’t want to have multiple sub-types of an object. For example, you may want to capture the Twitter handle of all new customers. Rather than creating a “Customer With Twitter” custom type, you’d just add a custom field to capture the Twitter handle.

## Step 4: Import and Verify Data

Next, import that data into commercetools. While commercetools does offer its own version of ImpEx, it’s easier to use some custom code to parse the CSV you exported from your legacy platform, extract the data you want, and import it into commercetools by calling the appropriate APIs. Java SDK users can use https://github.com/commercetools/commercetools-sync-java or else it’s simple to use the SDK of your choice (https://docs.commercetools.com/software-development-kits.html) as well.



Long-term, you'll still need to keep an up-to-date copy of your product catalog in your old platform because there are "hard" references to products, categories, etc. It's easiest to continue to push your raw product catalog data from your ERP, PIM, or other product catalog master to your current solution, as you have traditionally done. In addition to that, start also feeding that data to commercetools, and only allow your business users to further enrich that data in commercetools. The "shells" of un-enriched product catalog data in the legacy platform is just fine since the product catalog details are all being served from commercetools.

## Step 5: Build Custom Extensions

As the term "customization" already suggests, it's impossible to find a one-size-fits-all solution for all implementations out there. Instead, this little example should serve as a showcase to give you a rough idea of how you could proceed.

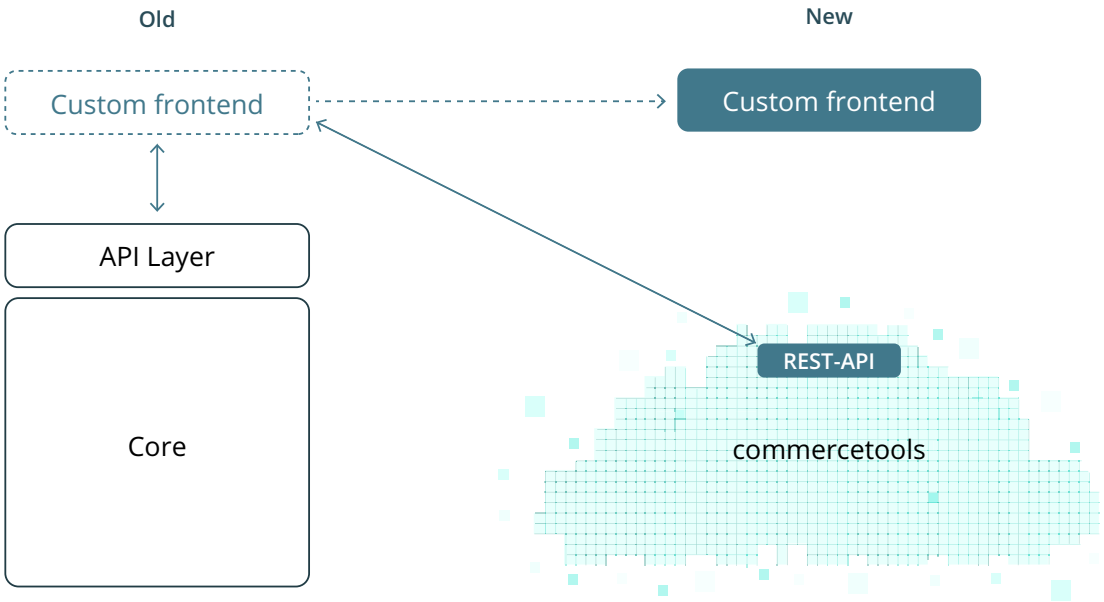
Let's say you managed to import your product catalog to commercetools and you have made sure that this catalog is regularly synced between the old and the new system. You can access them via the Products endpoints and work with them in any way you want – like displaying them in the Storefront which is otherwise powered by the legacy platform. In other words, you're overriding this part of the process by having commercetools deliver the data – a classic case of Martin Fowler's strangler pattern.

## Step 6: Migrate User Interface

The last item on your migration agenda is the user interface. The term "migrate" is a bit of a misnomer in this context, because in most cases you are facing a complete rebuild – just because monolithic commerce suites and commercetools are so fundamentally different. Especially if you have used something like a Hybris accelerator as the basis for your frontend, it's not possible to carve out and re-use the frontend code – because the storefront is tightly connected to your old platform's core.

| Monolithic platform | commercetools |
| --- | --- |
| Storefront application | Headless commerce platform |
| Accelerators for verticals | SUNRISE Quickstart template |
| API-layer covering a fraction of the core features | API-first approach, 100% functional coverage |

The only context in which a frontend migration might be possible is if you already have a custom frontend based on technology such as Angular or React and use an API layer in the old platform. In this scenario, the strangler pattern shown in the previous step can help with UX building as well. By identifying how the frontend sends data to and receives data from the current API layer, it is then possible to substitute the original endpoints with the ones provided by the commercetools platform.




# Conclusion

In this white paper, we have suggested a framework for helping you migrate your digital assets from your current, monolithic software stack to commercetools. As we have mentioned before, 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, most artifacts have to be built from scratch. The good news 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 your customers, you cannot have technical teams deal with maintenance and updates – instead, they need to build customer-facing features which generate real tangible business value.

commercetools
*Next generation commerce*

## About the author

Dr. Roman Zenner

Since 2001, Dr. Roman Zenner works as an author, consultant, and speaker in e-commerce. He has written several books on web shop software and regularly publishes articles in professional magazines and blogs. Dr. Zenner runs shoptechblog.de as well as the podcast ShopTechTalks. Furthermore, he speaks at conferences, teaches university classes, and moderates expert panels.

In his work, Roman focuses on next generation commerce technologies and explores what retail will look like in a post-web world. Since 2015, he is a full-time employee of commercetools GmbH (REWE Group), working as an Industry Analyst.

## 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 and meet new customer demands. It is the perfect starting point for customized microservices.

**Contact**
Europe - HQ
Adams-Lehmann-Str. 44
80797 Munich, Germany
Tel. +49 (89) 99 82 996-0
info@commercetools.com

Americas
American Tobacco Campus | Reed Building
318 Blackwell St. Suite 240
Durham, NC 27701, USA
Tel. +1 212-220-3809
mail@commercetools.com