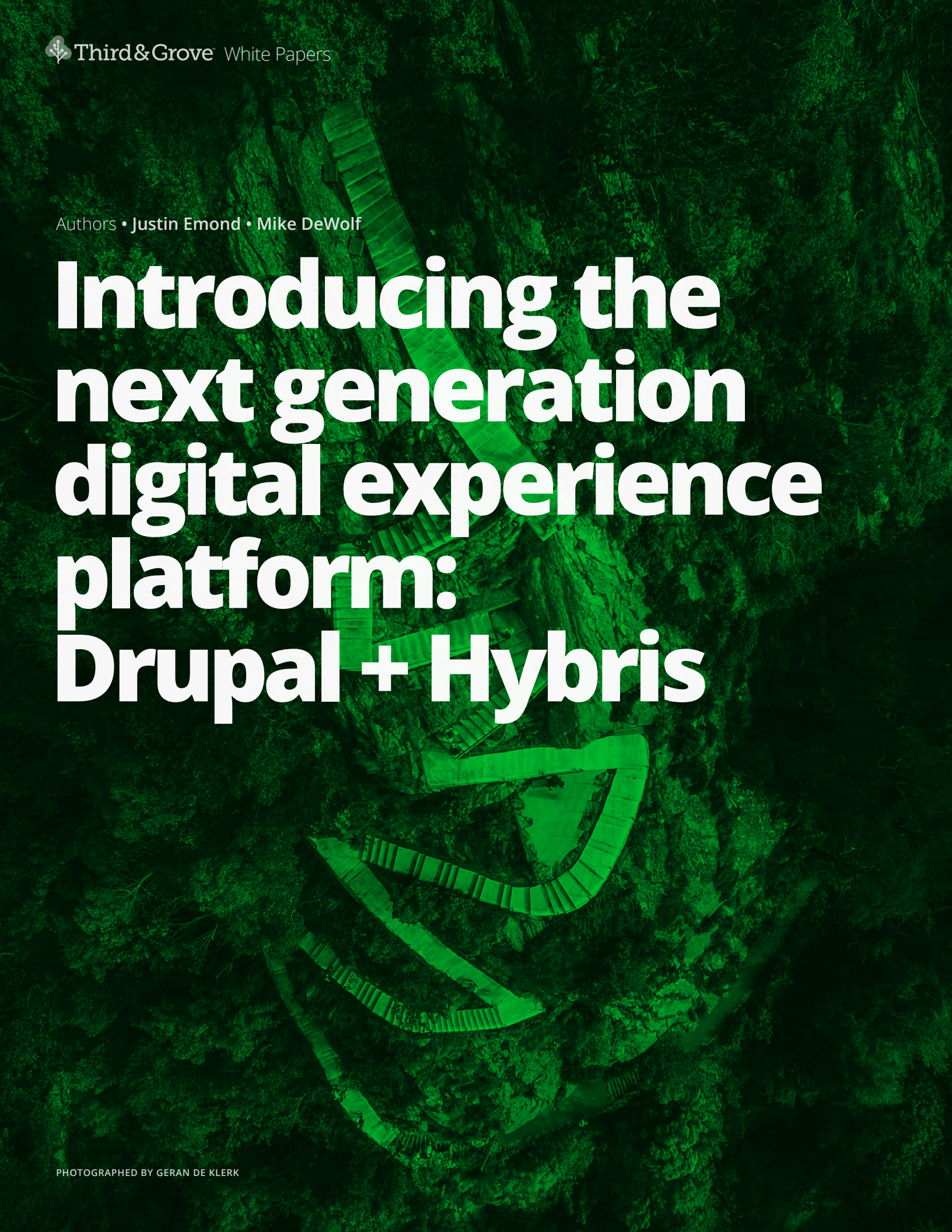
Authors • **Justin Emond** • **Mike DeWolf**

Introducing the next generation digital experience platform: Drupal + Hybris



PHOTOGRAPHED BY GERAN DE KLERK

Introduction

Hybris is a digital commerce platform used by some of the largest global brands in the world. With deep integrations into the **SAP** solution stack and the ability to handle scale and commerce complexity, Hybris is an powerful commerce engine for enterprise business.

However, because Hybris lacks flexibility in its theming engagement layer, as well as extensions contributed by the community to further its content experience capabilities, the platform struggles to provide the modern digital experience that consumers demand (an experience that drives revenue beyond cart optimization). While Hybris does provide strong support for digital commerce use cases, it fails to create engaging digital experiences.

Given the demands on today's business and marketing teams, as well as the expectations of a growing number of digitally native consumers, there is greater need than ever before for a platform that empowers the creative expression of digital experiences.

With the advancement of Hybris's core API architecture, it is now possible to combine the power of its commerce capabilities with the strength of Drupal's digital experience. With these two platforms, organizations can leverage the extensive frontend power of Drupal with Hybris' rock solid commerce backend. And because Hybris isn't replaced entirely, all of the connectors, integrations, and data flows built to support your digital commerce business remain intact.

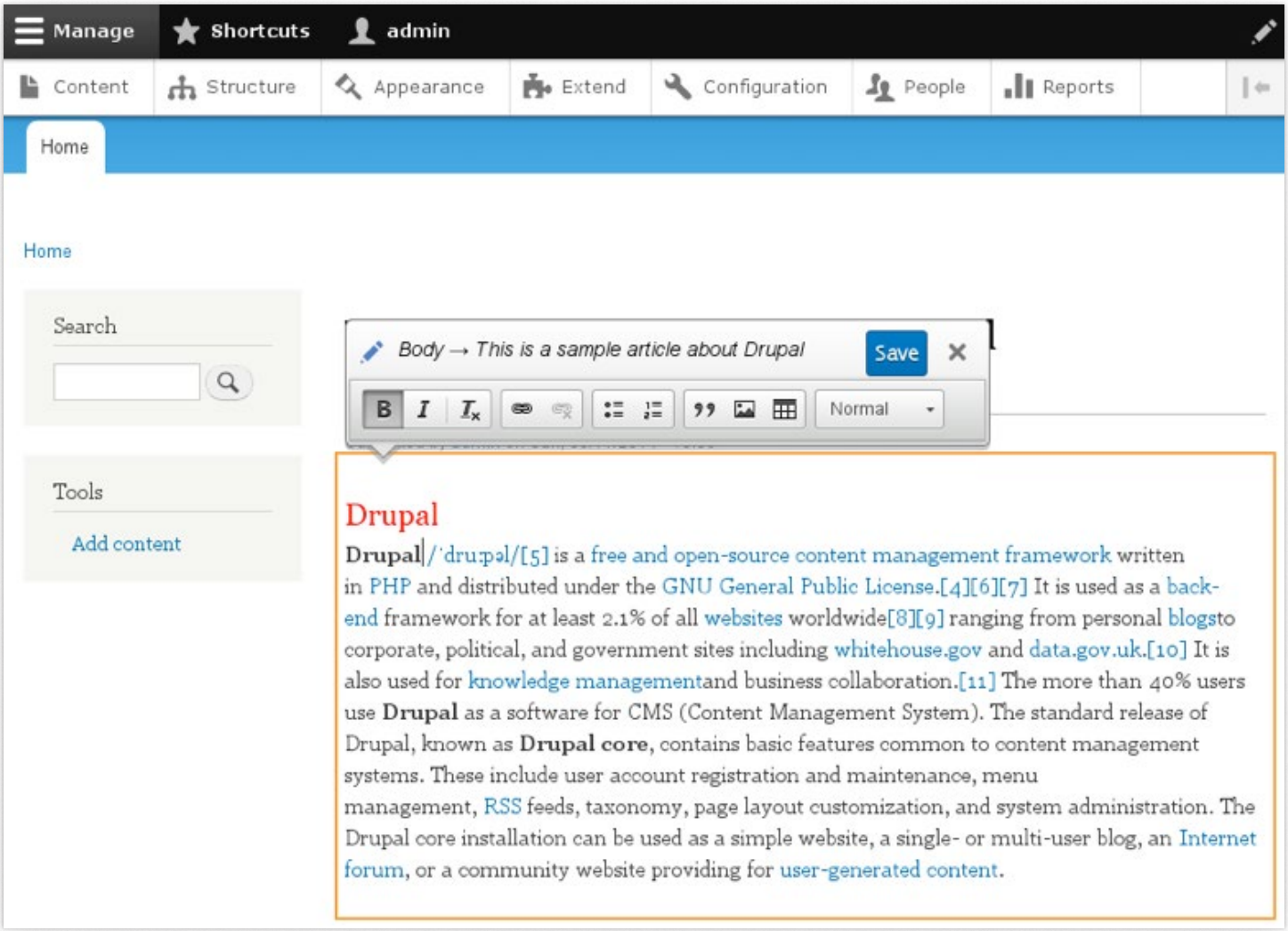
In this white paper, we will describe a technical approach to combining these two platforms, provide the tools for you to plan your own low-risk build, and share the lessons we have learned, so that your project can benefit from our experiences.

Key Features of a combined platform

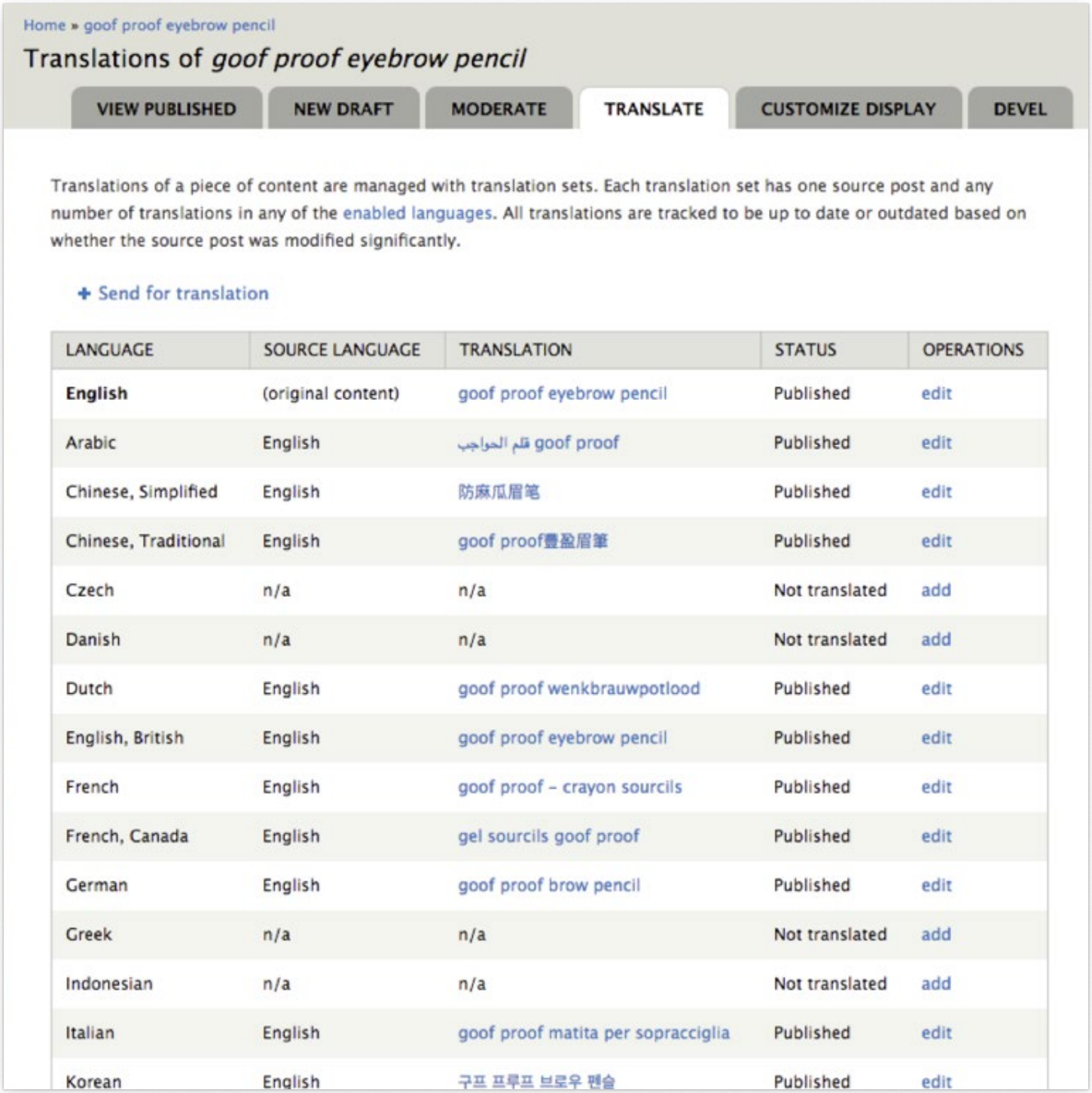
Combining Drupal and Hybris into a single digital experience platform provides a rich functionality that allows you to build engaging experiences. The benefits include:

+ Best-of-breed editorial workflow support, which empowers marketing teams of all sizes
+ Robust support for multiple languages and translations
+ Support for leading translation platforms, including **Translations.com** and **LingoTek**
+ Inline and in-place editing, making workflow easy for non-technical writers
+ Ability to leverage the universe of more than 10,000 free Drupal extensions to add functionality to the experience
+ Complete control over the checkout experience, no matter the number of pages
+ Full checkout capabilities, including address validation, PCI-compliant credit card capture, shipping integration, stored customer profiles, and more
+ Full support for Hybris promotions
+ Support for simple, configurable, and bundled products, with the ability to add support for any custom product type
+ Full profile integration with Hybris customer data, so there is duplicate information in Drupal
+ Support for unit and functional tests
+ Ability to deploy a content hub, where a network of localized sites integrate content and products from a central master site
+ Modern best practices tools Git, continuous integration, and release scripting

Drupal 8's in-place editing feature in action. No more scary back end interfaces, oh my.



The multilingual details for a specific product page. Each of these languages is editing using Drupal's powerful editing experience.



Home » goof proof eyebrow pencil

Translations of *goof proof eyebrow pencil*

VIEW PUBLISHED | NEW DRAFT | MODERATE | TRANSLATE | CUSTOMIZE DISPLAY | DEVEL

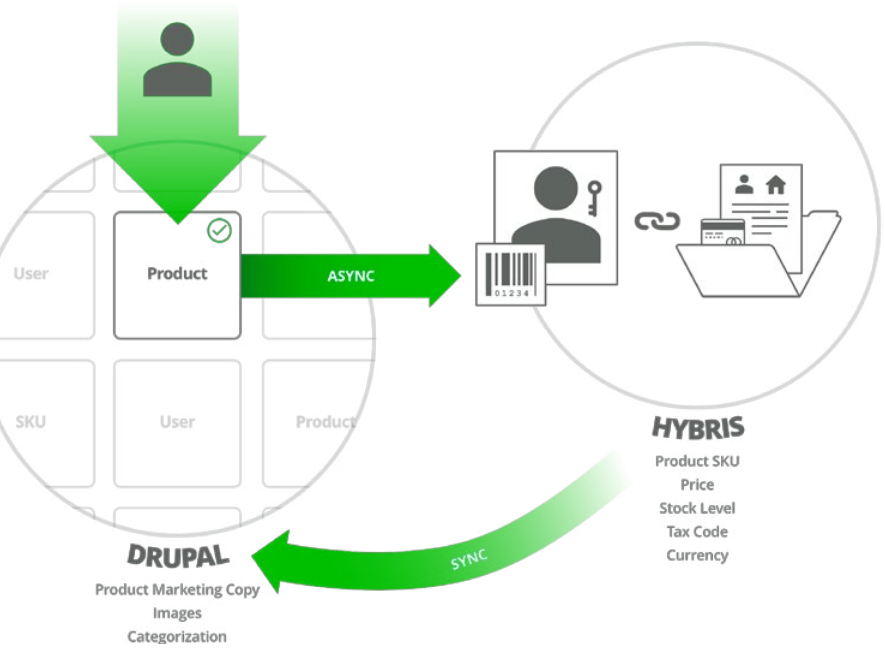
Translations of a piece of content are managed with translation sets. Each translation set has one source post and any number of translations in any of the enabled languages. All translations are tracked to be up to date or outdated based on whether the source post was modified significantly.

+ Send for translation

LANGUAGE	SOURCE LANGUAGE	TRANSLATION	STATUS	OPERATIONS
English	(original content)	goof proof eyebrow pencil	Published	edit
Arabic	English	قلم الحواجب goof proof	Published	edit
Chinese, Simplified	English	防麻瓜眉笔	Published	edit
Chinese, Traditional	English	goof proof豐盈眉筆	Published	edit
Czech	n/a	n/a	Not translated	add
Danish	n/a	n/a	Not translated	add
Dutch	English	goof proof wenkbrauwpotlood	Published	edit
English, British	English	goof proof eyebrow pencil	Published	edit
French	English	goof proof – crayon sourcils	Published	edit
French, Canada	English	gel sourcils goof proof	Published	edit
German	English	goof proof brow pencil	Published	edit
Greek	n/a	n/a	Not translated	add
Indonesian	n/a	n/a	Not translated	add
Italian	English	goof proof matita per sopracciglia	Published	edit
Korean	English	구프 프루프 브로우 펜슬	Published	edit

Understanding the technical architecture

There are two methods of integrating Drupal and Hybris: headless and side-by-side. In side-by side, Drupal renders all of the pages except for checkout pages, which are proxied to Hybris. Using the headless method, Drupal controls the entire front end, including checkout, and transactions are processed in Hybris via REST API.



What is the glass?

The glass refers to the platform that builds the web pages the end user interacts with. In the headless approach, Drupal is serving the glass—the end user never directly interacts with Hybris. The Drupal platform itself interacts directly with Hybris via Hybris' APIs on the user's behalf.

How to decide: headless versus side-by-side

There's no right or wrong approach to combining these two systems. Both headless and side-by-side architectures have pros and cons. By using your digital experience and commerce business requirements to drive the technical approach, you will minimize the total cost of ownership for your organization over time and avoid the unnecessary accumulation of technical debt.

To figure out the best approach for you, divide all of your major, high-level requirements into two categories: content requirements and commerce requirements. Now look at the number of requirements in each bucket. If there are far more commerce requirements, that may indicate a side-by-side approach is more ideal.

If the list is even, you need to look more deeply into the commerce requirements and understand which parts of the digital experience they impact. If you have commerce requirements that have significant interaction with the glass, lean towards a side-by-side approach; for requirements that are more backend-integration focused, go in the opposite direction. Sites that primarily have content requirements would also do well with the headless approach.

What is technical debt?

Technical debt is what happens when decisions made early in a project result in an accumulation of deferred effort. Technical debt eventually will have to be paid, it can make other phases of hte project more difficult (or, in extreme cases, so complex that it's not worth the effort). Typically, decisions that shortcut technical approaches for the sake of expediency or unsound technical decisions are the biggest contributors to technical debt.

The Pros and Cons

It's important to understand the pros and cons of each approach:

HEADLESS

Pros

- Highly secure, as the ecommerce platform can be off the public Internet
- One platform to theme instead of two (Drupal and Hybris themes cannot be shared)
- Does not require single sign on (SSO) between the two systems
- More robust content marketing platform for editors
- More powerful content workflow features

Cons

- Forward-facing Hybris extensions will assume Hybris is controlling the glass and may require additional refinements to work
- While Hybris has a robust, modern API architecture at its core, headless wasn't its original use case

SIDE-BY-SIDE

Pros

- Hybris extensions are usually not complicated by integration
- Maximizes the value of both platforms, as both are being used at their core use cases

Cons

- Twice the attack surface as headless (two platforms)
- Two themes must be built and maintained
- Breakdown of which pages should be served by which platform is not always clear during development
- Drupal needs full stack performance tuning, while Hybris needs only API-level tuning.
- Requires single sign on (SSO) integration

Architecture Decision Matrix

Use the architecture decision matrix below for a complete step-by-step guide to selecting the right Acquia Drupal + Hybris architecture platform.

	YES	NO
Is the product detail page complex, and will it be powered by many Hybris extensions?	Headless will increase the complexity of leveraging extensions involved in the product detail page. +1 for side-by-side	+1 for headless
Does the product display data or availability based on inventory fluctuate at a high rate?	Headless will generally increase the complexity of updating product display in real time for high rates of product data or inventory. status changes.	Does not impact integration approach.
Is payment gathered via modern, fully-hosted providers?	Does not impact integration approach.	Headless will require additional consideration for secure handling and compliance for protected payment method data. +1 for side-by-side
Is the checkout flow highly customized?	You may want to consider having Hybris host just the cart and checkout pages.	+1 for headless
Will the site have a blog?	+1 for headless	+1 for side-by-side
Will the site have many content-driven landing pages?	+1 for headless	+1 for side-by-side
Do you need faceted search?	Drupal has better support for indexing, formating, and parsing your site's content—product, marketing pages, and otherwise—in search results. +1 for headless	Does not impact integration approach.
Will there be a complex logged-in experience?	Drupal 8's framework for logged-in experiences is more efficient to extend and build on than Hybris's. +1 for headless	Does not impact integration approach.
Are you using content personalization?	+1 for headless While market-leading personalization tools like Acquia Lift and Optimizely work with both platforms, having one platform to glass will simplify the integration.	Does not impact integration approach.
Do you need editorial workflows for marketing content?	+1 for headless Drupal is built primarily to be a tool for editorial teams. This is where it excels.	Does not impact integration approach.
Are there CRM integrations?	Drupal has strong support for integrations with leading CRMs like Salesforce. Hybris does not have strong support other than for its own product family (SAP). +1 for headless	Does not impact integration approach.
Are there commerce ERP integrations?	Integration approach does not impact ERP integration complexity.	Integration approach does not impact ERP integration complexity.

Managing SKUs: architecture best practices

Hybris and Drupal each have their own way of handling data. Both are well adapted to modeling product, user, taxonomy, and even customizable data structures. From a software engineering perspective, it is helpful early on in the build to decide which data will be managed in Hybris and which will be managed in Drupal.

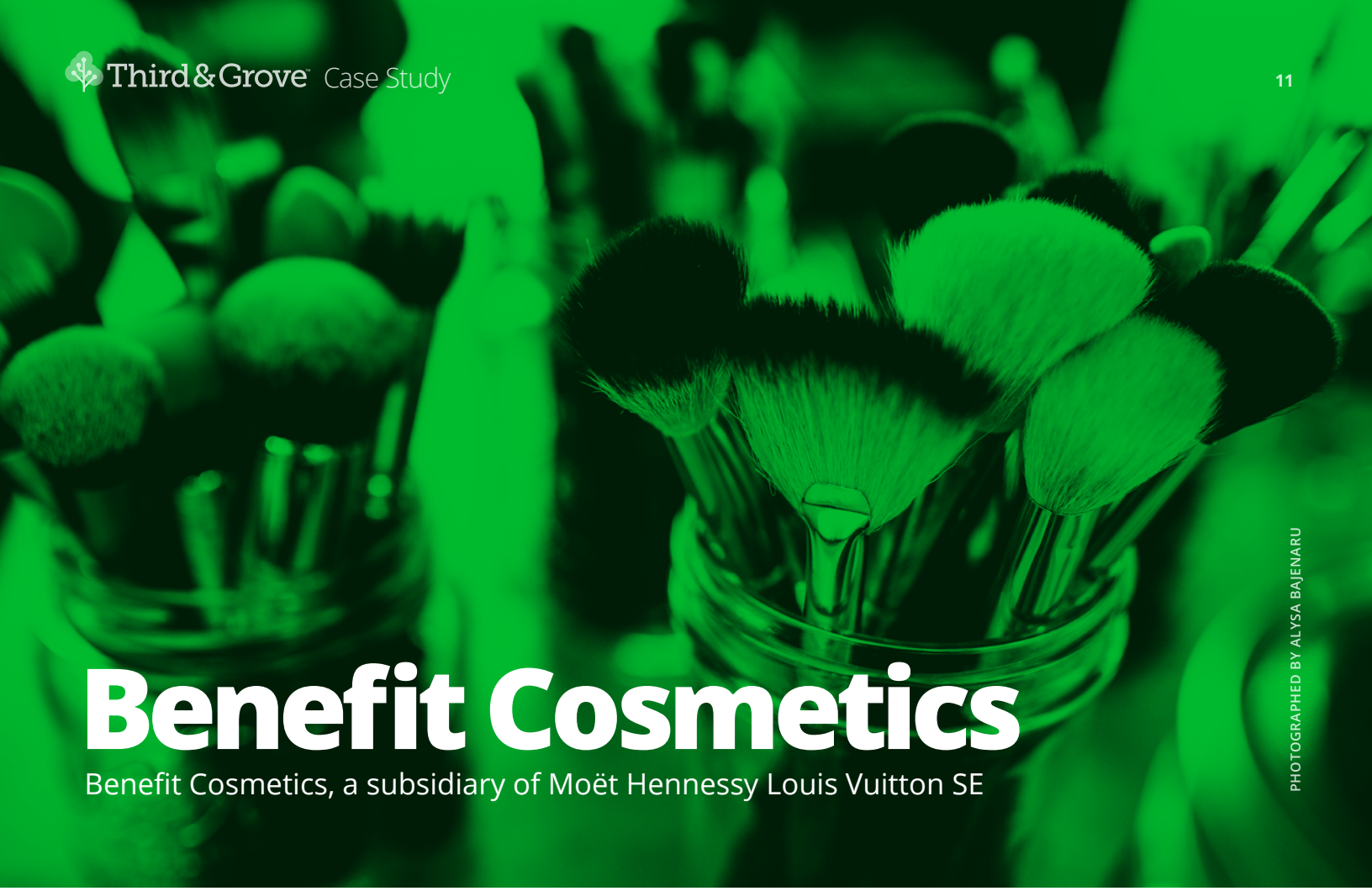
We often have found that the ideal split between the two platforms mirrors an organization's marketing and fulfillment team structure. A good way of dividing it is to picture the fulfillment team only using Hybris and the marketing team only using Drupal. Along these lines:

- A fulfillment team needs their platform to operate as a product and account management utility and may need close integration with CRM, ERP, and/or PIM systems
- A marketing team needs their platform to be dynamic, rapidly customizable, and even personalized to the visitor, while integrating with various analytics platforms and lead funnels (Marketo, Eloqua, Salesforce etc.)

The most common split for source of truth is to put critical ecommerce data like user addresses, order histories, price, SKU, and stock in Hybris, and all frontend product marketing and display data in Drupal. Done right, the two sides of the business should only need logins to a single system.

The end result is a system that isolates ecommerce data, giving the marketing team free reign over a rich, content-driven Drupal website. Drupal updates price, stock, and other fulfillment-related attributes from the Hybris backend both synchronously and in real time in response to edits.

So how does this look in the context of Drupal content types? In order to structure a headless ecommerce experience in Drupal, we make a product node type that is edited in Drupal and operates as the presentation entity for the product view. We then add custom commerce SKU entities (associated via entity reference) to make product attributes that are sourced from Hybris available to the presentation layer. Using cron routines and drush commands, we sync SKU data between the two systems. Structuring SKUs as custom entities allows us to create different bundles to handle the different front end processes necessary to support the the data and logic behind different product cockpit attributes.(ie configurable, bundleable, grouped, downloadable, custom etc.).

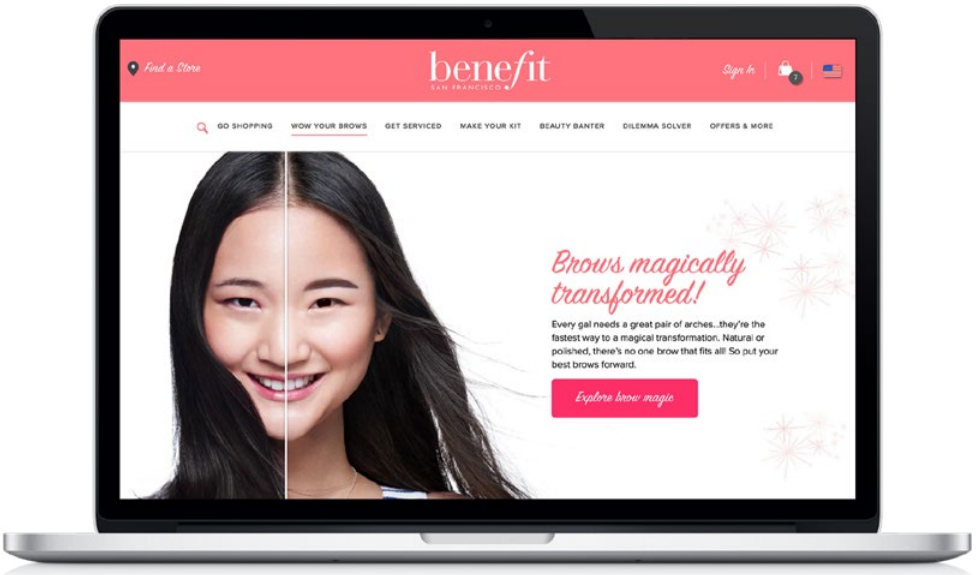


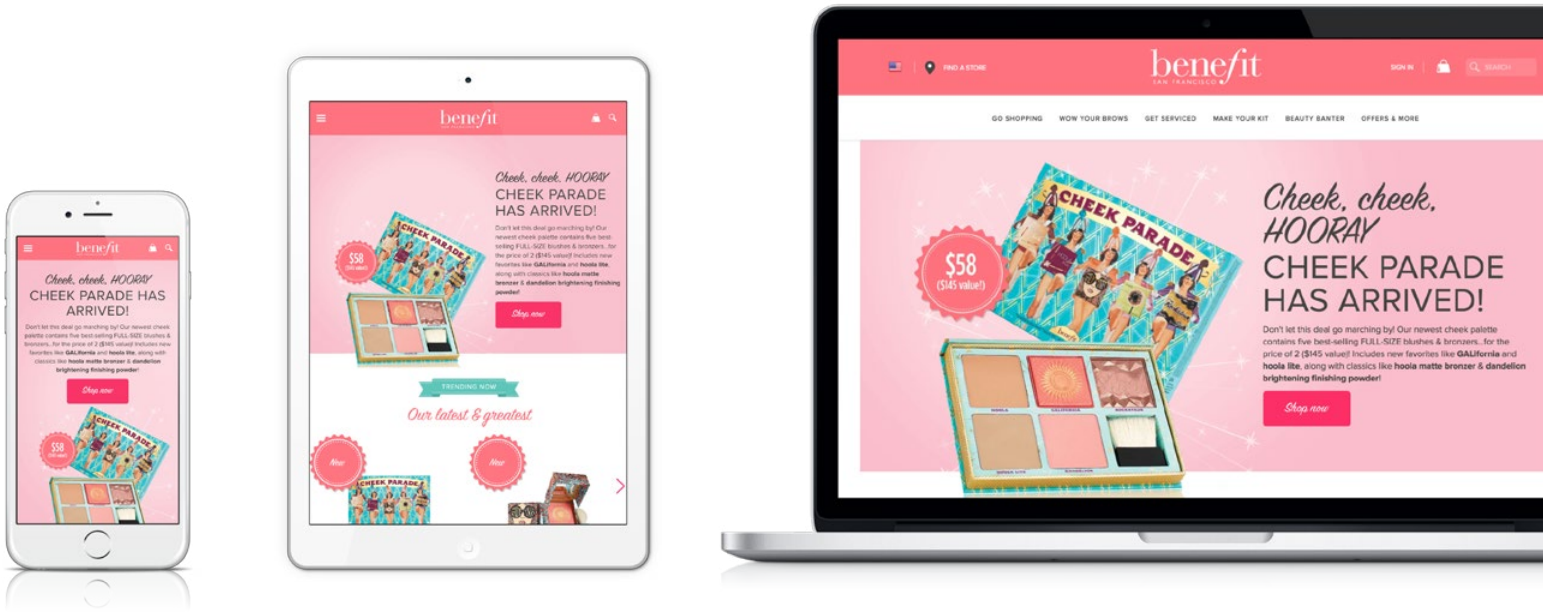
PHOTOGRAPHED BY ALYSA BAJENARU

Benefit Cosmetics

Benefit Cosmetics, a subsidiary of Moët Hennessy Louis Vuitton SE

The rebuilt Benefit Cosmetics site is flush with features and sophistication, yet still an effortless experience for its customers—much like the beauty products created by the fashionable, San Francisco-based brand.





The Dilemma

Benefit Cosmetics is known for its irreverent voice, radiant personality, and unique, dilemma-based shopping experience, but the back end of its site was due for a makeover. The company wanted to seamlessly integrate its order fulfillment system, drive sales by enabling its tech savvy customers to shop from any device, and scale its site and business internationally.

Our Approach

We focused on creating a dynamic experience for the flagship U.S. site, while developing a hub and-spoke model for the international rollout, complete with translation workflow integration. Our solution made it simple for the company to run dozens of online stores in countries scattered across the make-up-hungry globe.

The Technology

Ecommerce / The Benefit team already had a robust online store managed with Hybris, SAP's enterprise ecommerce solution, but as the marketing department's needs became more sophisticated, the existing CMS was no longer up to the task.

Drupal / Benefit wanted a platform that would enable its marketing team to manage content easily, offer strong support for running sites in multiple languages, handle the peak traffic load of a major cosmetics brand, and allow for heavy integration with an enterprise back end system.

Custom Front-End / The site needed to be responsive, flexible, and interactive, and above all, provide a seamless shopping experience. Benefit's marketing and content teams are now able to maintain the brand's unique design aesthetic and voice while customizing content for user needs.

Multi-site, Multi-Language / The new platform will eventually power commerce and content sites in more than 40 countries. To ensure the sanity of the translations workflow, we completely rewrote the Translations.com Drupal module and contributed our work back to the open source community.

The Transformation

The new site launched in January 2016, a beautiful blend of content and ecommerce. Drupal handles the content and rendering, while Hybris takes care of the ecommerce and back office integration.

Making it scale

The beautiful part about this integration is if you've followed the technical advice in this white paper about how to structure your entities, you'll already have a near perfectly delegated cache model. Product pages and marketing content on the site shouldn't need to expire frequently, and since we are using Drupal 8, we can manually purge pages when necessary. Because we are using Drupal entities, we can cache these in whatever datastore we are using on our web server (Memcache/Redis). We can also opcache our PHP code as it compiles, resulting in the delivery of a tight, cacheable package from our application layer.

We recommend placing a varnish cache in front of the entire site to speed things up and avoid unnecessary Apache load. We also recommend putting a CDN in front of the entire system for Geo-availability, performance, and security.

Usually cache exclusion rules for an estore setup like this are relatively simple. Only shopping cart paths need to be excluded.

Five Must Do Tips for Drupal Hybris Integration Projects

1. Choose Sources of Truth

Decide which system will be the source of truth for each data point. Usually the breakdown will mirror the organization's marketing and fulfillment team structure, with marketing data controlled by Drupal and business and fulfillment data controlled in the ecommerce platform. Make auew users only need logins to a single system.

2. Model Hybris Products in Drupal

Make a product node type which that can be edited in Drupal and operates as the presentation entity for the product view. Add custom Commerce SKU entities (associated via entity reference) to make product attributes that are sourced from Hybris available to the presentation layer. Write cron and drush routines to sync SKU data between the two systems. By separating SKU from product view, you give the marketing team full freedom over product content without jeopardizing any ecommerce-related data.

3. Create a loose coupling between the ecommerce and marketing components of the website

Consider the invariable eStore outage when structuring Drupal's relationship with the Hybris API layer. Plan what the end user experience will be during an outage and make a setting to turn off ecommerce accessible via the Drupal GUI. That user experience can be as simple as a redirect on checkout paths, or as elaborate as hiding the add to cart buttons on all product pages and removing links to cart from page templates.

4. Keep Price, Tax, and Shipping in Hybris

Hybris should always maintain supremacy over price and other data essential to the checkout transaction. What a customer sees in their basket, their cart, or during checkout should always be an accurate reflection of what the ecommerce backend is saying.

We have a rule for these integrations: There is no math in Drupal. We sync price fields as strings and display price totals during checkout via live API calls. We remove duplicative logic by taking all math out of Drupal, and ensure that there are never discrepancies between what a user sees on the checkout frontend and what they are billed for.

5. Consider Using a Conductor Platform

A conductor platform is a message bus system that handles non checkout-essential checkout transactions/integrations (CRM updates, email lists etc.) so that if one of these systems goes down, the eStore still functions. Using a conductor improves fault tolerance by removing non-essential http transactions from checkout and deferring them to a queue system that will store the requests until the operation succeeds. If a CRM update fails because of an outage, the conductor tries again later. Successful checkout is mission critical, so it makes sense to decouple non-essential operations from the logic-driving order completion.

Getting Started

To get starting integrating Drupal + Hybris, download the Hybris integration connectors we released on Drupal.org: **drupal.org/project/hybris**

Be sure to read this entire white paper, and keep an eye on our blog, as we are constantly sharing new information on using these two platforms: **thirdandgrove.com/articles**

Third & Grove can help with your project

Our experienced engineering team has implemented Drupal and Hybris at scale for a variety of clients. Our hard-earned experience makes our team efficient and a very low-risk partner to customize the integration.

We have a very flexible engagement model and are available to help with refinements to the integration, building the entire platform Drupal + Hybris platform, or augmenting your internal team with architecture guidance and leadership on the integration process.

The importance of using a conductor

Acquia Drupal and Hybris are both enterprise systems that have a proven record of scaling to handle large order volumes and traffic, authenticated or otherwise. However, a modern digital commerce experience involves a variety of integrations with third-party and internal systems for order processing, financial reporting, and customer service.

Conductor platforms are a critical component for ensuring fault tolerance in Acquia Drupal + Hybris integrations. A conductor stands between Hybris and every integration system that is not a checkout blocker, and it holds requests until the system is available. Data doesn't slow as it's sent out of the store, and if third-party systems go down, they won't take down the store. In many cases, only the payment system is a blocker, and you can dramatically reduce your single points of failure.

For example, if your store needs to send order information to Salesforce so that your customer support team has the information it needs, a delay in sending that order to Salesforce of a few seconds is trivial and doesn't impact business. Salesforce can be down— the data sending request can fail for any reason, really—and your digital commerce stays online.

About the authors

Justin Emond

CEO

Justin is the founder and original CTO of Third & Grove, a technology focused digital agency in Boston and San Francisco. Justin has worked on both sides of the technology world—on the inside of companies and as a consultant. Prior to Third & Grove, Justin served as Senior Digital Project Manager at a boutique agency in southern California and as adjunct faculty at the top-ranked USC Viterbi School of Engineering. He is the author of **Pro Web Project Management (Apress, 2011)**, a pragmatic guide to managing digital projects.

Mike DeWolf

Director of Engineering

Mike began his career in software engineering as a web freelancer, then spent several years in the internal technology department at Mountford Media, a content generation company based in the UK. Mike knows Drupal and Wordpress inside and out, and loves to build custom applications. In his spare time, if he's not wrangling javascript, he's probably in the great outdoors.

PHOTOGRAPHED BY ALYSA BAJENARU

About Third & Grove

Third & Grove is an independent agency of innovators, designers, and engineers. We work directly with incredible organizations to build complex systems and innovative digital experiences in technologies like Acquia Drupal and Hybris.

Third & Grove a full-time team of 40 people and two offices in Boston and San Francisco. Our key discipline is delivering engaging digital experiences. We achieve this by breaking rules while remaining technologically sound.

Our company is organized respectively into two groups:

- A strategy group that is responsible for digital strategy, user research, UX, and design.
- A technology group that is responsible for code architecture, testing, and delivery.

TAG's strategy group consists of solution strategists, business analysts, content strategists, social marketing experts, SEO experts, visual designers, and interactive designers.

This team is responsible for interpreting a client's needs and delivering a solution through analysis, research, user testing, and forward-thinking design.

Our proven technology group is made up of architects, developers, and quality assurance resources. Together, we have delivered mobile and web solutions (on spec, on time and on budget) for Fortune 500 companies and startups alike.