Coveo for Commerce

Implementation Guide

December 2022

Contents

Introduction 4
What is Coveo for Commerce 4
Coveo Commerce Architecture 5
Planning the Implementation 6
Team Expertise 6
Getting Started 7
Tracking Data 8
Standard Events 8
Commerce Events 8
About the PermanentId Field 9
Make Products Discoverable 10
Catalog Architecture 10
Indexing Your Catalog 13
Exploring and Structuring Indexed Content 14
Catalog Manager 14
Adding Non-Catalog Content 15
Serving Content 16
Touchpoints 16
Understanding Search Hubs 16
Using a Coveo Framework 17
Creating a Global Search Page 18
Creating Listing Pages 19
Adding Product Recommendations 19
Styling 21
Templating Results 21
Tracking Clicks 21
Sending Context Data 22
Search Engine Optimization 22
Localizing the Interface 23

Improving Relevance 24
Machine Learning Models 24
Query Pipelines 26
Pitfalls and Misconceptions 28
Search-As-You-Type 28
Using Ranking Tools Without Data 28
Wildcarding 29
Implementation Checklist 30
Coveo Resources 31

Introduction

Coveo is a highly scalable, highly extensible, cloud-based search, recommendations, and personalization platform. It uses robust machine learning technologies to provide the most relevant experiences possible to virtually unlimited use cases.

This document is intended for anyone planning to use Coveo to power their commerce search experiences. It guides you through the major steps and design questions required when developing with Coveo.

This guide does not intend to replace the **product documentation**, but shares best practices for a successful implementation. For a step-by-step learning experience, see the **Level Up platform**.

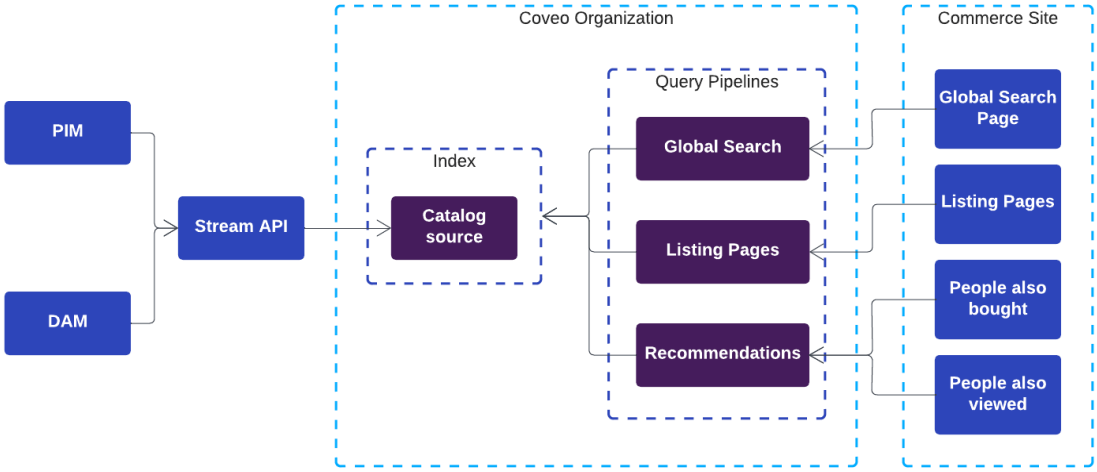
What is Coveo for Commerce

Coveo for Commerce boosts marketplace performance with relevant search and personalized buying experiences, using data from every user interaction to fuel Coveo Machine Learning (Coveo ML) and displaying products they're likely to buy based on site behavior, their profile, similar user habits, frequently added to cart items, and more.

The main difference between Coveo Commerce and Coveo for Websites is the presence of a purchase mechanism. If you have a website that displays all your products but doesn't allow users to purchase items directly from the website, or only allows them to purchase them through a process that cannot be tracked (e.g., by asking for a quote that would then fall in the hands of a Sales team), then your use case would be considered a "digital brochure," a Coveo Website use case. If your site has a cart and a full shopping experience, it should be considered a Coveo Commerce use case.

If your project is more of a digital brochure, the **Coveo Platform Implementation Guide** would be better suited to help you during your project.

Coveo Commerce Architecture



The Coveo Commerce architecture can be summarized as such:

- Content from your product information management system (PIM), digital asset management system (DAM), and other repositories is aggregated and formatted in a way that can be read and understood by Coveo. Each item generated at this stage should have the appropriate metadata required by Coveo.
- The items are then sent via the Coveo Stream API to a Catalog source in your Coveo organization. The source takes care of reading the appropriate metadata and adding each item to your Coveo index.
- When a user types a query in a search box on your site, it goes through the Global Search pipeline, where results are returned according to the user's profile and past visits to provide the most relevant content.
- When a user accesses a listing page (e.g., a list of all products from a specific category), the request goes through the Listing Pages pipeline, and returns all products of that category, ordering results in a similarly relevant manner.
- When a user gets to a page where a component recommends them other products, a request goes through the Recommendations pipeline, and based on the context where that component is as well as information about this user and other users' purchase history and visits, Coveo returns products that are relevant to the user at that moment.

Planning the Implementation

A successful Coveo Commerce implementation requires a few core components:

- Good and accurate index data
- Complete front-end integration
- Robust user interaction data collection

Without one of these core components, a Coveo Commerce implementation will struggle to provide relevant products to your customers. With all three, you can expect Coveo's Machine Learning capabilities to greatly improve the user experience on your site.

Team Expertise

To bring a Coveo project to completion, you need people with a certain type of expertise. You want at least one person who can perform each of the following tasks:

- Design an architecture to the project so every aspect is aligned and business requirements are all taken into account
- Extract data from the PIM and other catalog repositories
- Take the extracted data and create a mechanism to format and push the content to Coveo via API's
- Build front-end components and pages using either the **Headless** or **Atomic** frameworks from Coveo
- Send analytics data from the front-end to improve relevance
- Design and build ranking rules such as promotions on the Coveo platform

Of course, some of those tasks can be assigned to the same person, and some may be distributed to more than one. However, all of these tasks need to be covered by at least one person. Additional tasks may also be needed, such as QA testing, post-implementation usage analysis, more robust project management, and more advanced business analysis.

Getting Started

Before you can start working on your Coveo implementation, you need a Coveo organization.

If you are a new Coveo customer, you should have been provided with at least a sandbox and a production organization. If you did not, please contact your Coveo representative or the Coveo administrator in your organization.

If you are not yet a Coveo customer, you can create a trial organization, which gives you the Enterprise features of Coveo for 30 days. For more information, please fill out **this form**.

Coveo currently offers organizations in three main regions. Depending on your region, the link to log in to the Coveo platform is different:

- US East: **platform.cloud.coveo.com**
- Ireland: **platform-eu.cloud.coveo.com**
- Australia: **platform-au.cloud.coveo.com**

By default, trial organizations are created in the US East region.

Tracking Data

A Coveo Commerce implementation requires good data tracking to be relevant.

There are two types of events to track: standard Coveo events, and Commerce-specific events.

Standard Events

Standard Coveo events refer to events that happen on Coveo components. This is typically limited to what the user searches, and which results they click on. The Automatic Relevance Tuning, Query Suggestions, and Dynamic Navigation Experience Machine Learning models require both those events to return data. These events are typically sent from the search components themselves. However, you most often want to add additional dimensions to allow Machine Learning to better learn from those events.

The **Serving Content** section covers how to send those events.

Commerce Events

Commerce specific events refer to events used to add more context to the Coveo specific ones. Often, those events are triggered outside of Coveo components. They include events for add to cart, page views, and purchase, among others. The different Product Recommendations Machine Learning models require those events to return data. These events are typically sent using the **coveo.analytics.js** script.

Commerce specific events can and should be tracked before the full Coveo experience is implemented. This is because Coveo models can start learning from live events even before the implementation is done, so that the day you go live, you already have healthy and relevant data and Machine Learning models.

To learn how to send the appropriate data, refer to the **Commerce Data Health Implementation Guide**.

About the PermanentId Field

When tracking commerce events, you are asked to send a field and a value, so that Coveo can know which specific page or product was consulted. The best practice is to use the field **permanentid**, and to send the product SKU for its value. Coveo's Machine Learning models will then be able to use the SKU from the **permanentid** field to link your catalog with your analytics data.

As the name implies, the **permanentid** value is expected to be permanent and unique for an item. Most of the time, SKU's are unique and permanent to a product.

For more information on the **permanentid** field, see **About the PermanentId Field**.

Make Products Discoverable

Good and accurate index data is the foundation when building a relevant experience. This section covers how to add catalog data to your Coveo index.

Catalog Architecture

To add your products in a Coveo index, you need to create a catalog source in your Coveo organization.

A catalog can contain three main objects: products, product variants, and availability indicators.

Differentiating Products and Variants

The difference between a product and a variant can be subtle, especially since it's common to group products together. Product grouping will be covered in the **Adding Product Grouping and Variants** section.

Products are what are shown to users on a results page, while variants are not shown, but are used to determine the different ways a product can be available.

The information displayed on the result list needs to be available in the product attributes. For example, price, brand, and description need to be part of the product data for them to be displayed in the results.

In a **catalog with variant data**, users search for products, and then select a variant to purchase. Variants are never returned as search results. Instead, they provide additional **metadata** on a parent product, which can be used for facets and sorts controls.

For example, let's take a t-shirt that comes in different colors (yellow, orange, and black) and different sizes (S, M, and L). The three colors would be three different products, which would be grouped together when serving the content. The different sizes would be variants, which are used to compute the filters a user can select on the search page.

Availability

Availability determines whether a given user can purchase a given product or variant. An availability can be a store inventory, a price list, or anything that controls which user has access to certain products or variants. In your typical B2C use case, it is mostly linked to physical stores, for example if your products are available for in-store pick up or local delivery. For the sake of clarity, this guide will refer to availability indicators as stores.

When using availability, stores need to be indexed as individual items, with the fields **storeid** and **availableskus** populated. Additionally, indexing latitude and longitude is recommended if you intend to leverage geolocalization.

The **availableskus** field should contain the id (or SKU's) of all variants or products available at that store. At query time, Coveo will use the SKU's from this field to know which products to return to the user.

Mapping Standard Fields

To get the most out of your implementation, you are required to use a standardized set of fields. The values from these fields are then used by the Coveo Machine Learning models to create product vectorization and perform better content analysis.

Additionally, you need to make sure that the data you are sending is normalized. For example, if you have a category of products called T-Shirts, you want to make sure all T-Shirts are sent with the same value in that field. You want to avoid sending different values like *T Shirt*, *tshirt*, *t-shirts*, when they should be considered the same.

To know more about standard fields, see **Standard Fields**.

Fields Options

Fields dictate how metadata should be stored and used. Fields can be of several different types, related to the programming types: **strings** (text), **integers** or **decimal** (numbers), or **dates**.

Fields also have options that you can change. The most important field options are:

- **Facet** — Lets you use the field for facet filters.
- **Multi-value facet** — Treats all values separated by semicolons as individual values, as opposed to one long string.
- **Sort** — Lets you use the field for sorting results. With string fields, sorting is alphabetical.
- **Free-text search** — Lets Coveo know that users should be able to search for content inside of that field, and return the result even if those words are not inside of the body of the document.

Make sure you only enable the appropriate options for the appropriate fields. Enabling an option when it is not needed may affect performance and/or relevance.

Source Languages

Coveo is a multilingual index which detects user language and adjusts relevance accordingly. The full list of **supported languages** is available online.

You should always create one source per language to establish a unique representation of the products within each source. This way, Coveo Machine Learning can provide a personalized experience between markets.

Indexing Your Catalog

The best way to get your products and variants in your Coveo index is to use the Stream API to add them to a Catalog source.

This process requires you to create a JSON representation of your products and variants, along with all the necessary metadata, and stream it through the Coveo API into your Coveo organization.

For more information on this topic, see **Index Commerce Catalog Content With the Stream API**.

Catalog Configuration

Configuring the catalog is the way you can tell Coveo how to organize your products, variants, and availability indicators together. To learn how to configure your catalog, see **Create a Coveo Commerce Catalog**.

Updating Content

When you want to update content in your catalog, for example when a price changes, or when a product or variant stops being available in a location, you should leverage the Update API to perform a partial document update.

However, if you want to add new products or variants, or delete existing ones, you should use the full document update method.

For more information on how to do that, see **How to Update Your Catalog**.

Exploring and Structuring Indexed Content

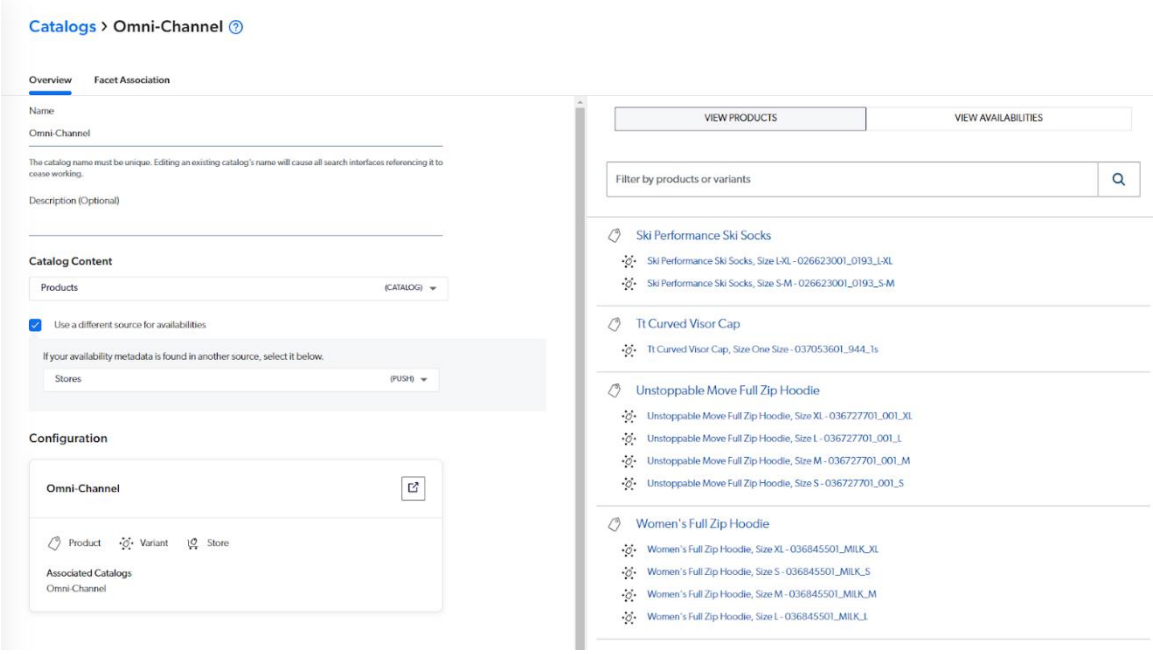
The **Content Browser** displays objects from all catalog source data in a single unified interface. It's the best tool to validate if your data has been properly indexed.

Tip: Newly indexed documents may not immediately appear in the Content Browser as they might still be processing. The View All Content option bypasses item/source security, allowing you to explore content you wouldn't otherwise be explicitly allowed to see as a user. Changing the pipeline is another way to ensure a filter is not removing items.

Catalog Manager

Once you have the data in the index, you are ready to **Create a Coveo Commerce Catalog**. The Coveo Catalog Manager in the Administration Console will help you build your catalog, which maps the relationship between your objects.

For this example, let's assume you have a single channel catalog structure composed of products and variants only.



This Coveo Catalog re-creates the product/variant relationship during the query execution. If you have a catalog with only products, the Coveo Catalog will still be useful, as it's used by Coveo Machine Learning models to map product data with user behavior.

Adding Non-Catalog Content

Coveo provides a unified index, meaning that it can index content from many different places (e.g., a website, CMS, or CRM software) using **different connectors** and store it in a single index. However, when serving that content, be aware that you can either show the content of a one catalog or anything outside of your catalog.

Serving Content

Once you have indexed your products in Coveo, you are ready to serve that content to users. This is done by creating a **search interface**.

Touchpoints

There are three main touchpoints where you will typically be serving content with Coveo: the global search page, listing pages, and recommendation components.

The global search page is the page reached when a user searches for something in the search box in the header of your site. It is the most full-featured search page, leveraging facets, sorting options, and other filtering methods.

Listing pages are typically reached via navigation, and normally present all products from a specific category. It may include facets and sorting options, but typically doesn't include a search box of its own.

Recommendation components present additional content to users without feeling like a search page. These include recommendations of products based on what the user has viewed (e.g., similar products), on what they have purchased in the past, on what they have in their cart, or on what people like them previously viewed or purchased.

Understanding Search Hubs

To know where a query is coming from, Coveo relies on the search hub parameter.

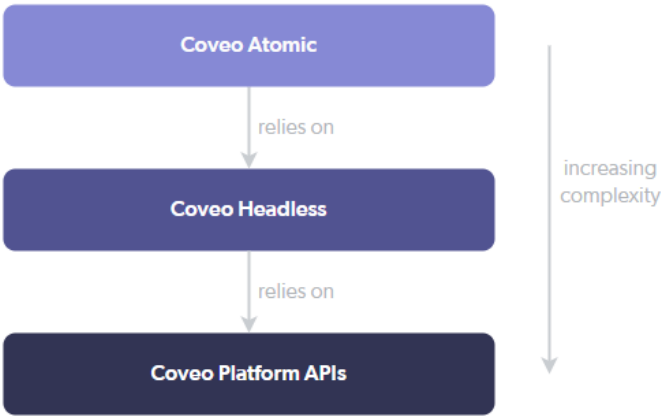
While search hub names are used by Coveo Machine Learning, they are also used by analysts looking at the analytics in your organization to determine how users are leveraging search. When naming search hubs, make sure you use human-readable values.

Make sure your search hub name starts with the name of the site or brand, followed by its purpose: **Search** for the global search page, **Listing** for listing pages, and **Rec** for recommendations. For example, **ACME Search** and **ACME Listing** are perfect search hub names.

The specific listing page is determined by a different parameter, called **tab**. Setting tabs will be covered in the **Creating Listing Pages** section.

Using a Coveo Framework

Coveo offers two frameworks that interface with the Coveo API's to create search pages: the Coveo Headless library and the Coveo Atomic framework.



Using the Coveo Headless Framework

The **Headless library** is a Redux-based tool set for developing your own search UI component libraries, acting as a middleware with the API's. It typically involves two main building blocks: the engine, which manages the state of the search interface and communicates with the Coveo Platform, and the controllers, which dispatch actions to the engine based on user interactions. You can learn more in our **Coveo Headless Tutorial**.

Using the Coveo Atomic Framework

Coveo Atomic is a web component library built on top of the Headless framework that allows for quicker assembly, featuring pre-built components. These web components can be used with or without existing frameworks.

Learn more about **Coveo Atomic**.

Creating a Global Search Page

Once you have decided on your framework, you can start creating your search pages.

The global search page should be its own page, not a panel that appears in the search box. You should also leverage different facets, so that users can more easily drill down on what they want.

Creating the Global Search Box

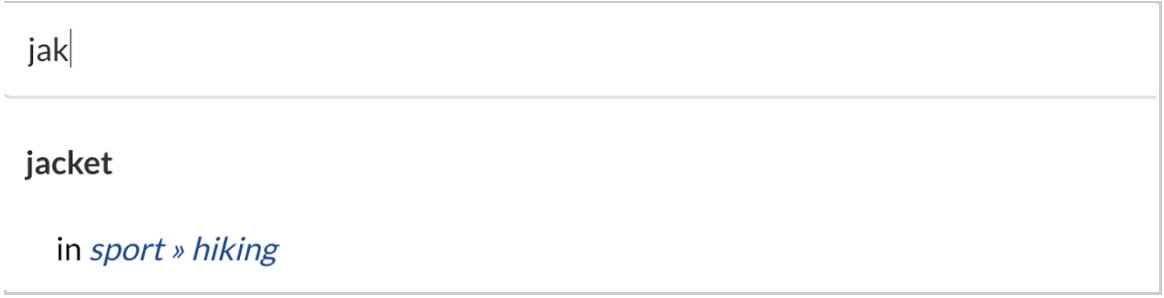
The global search box refers to the search box in the header of your site, which redirects to the global search page.

Most of the time, the global search box does not perform queries directly. Instead, it suggests queries (through machine learning) to users based on the letters they typed and, on their profile, and then redirects them to the search page with the selected or typed query. For Coveo to provide the appropriate query suggestions, you need to make sure the search box and the page it redirects to share the same search hub.

There are two additional features you might want to consider for your global search box: category suggestions and product suggestions.

Category Suggestion

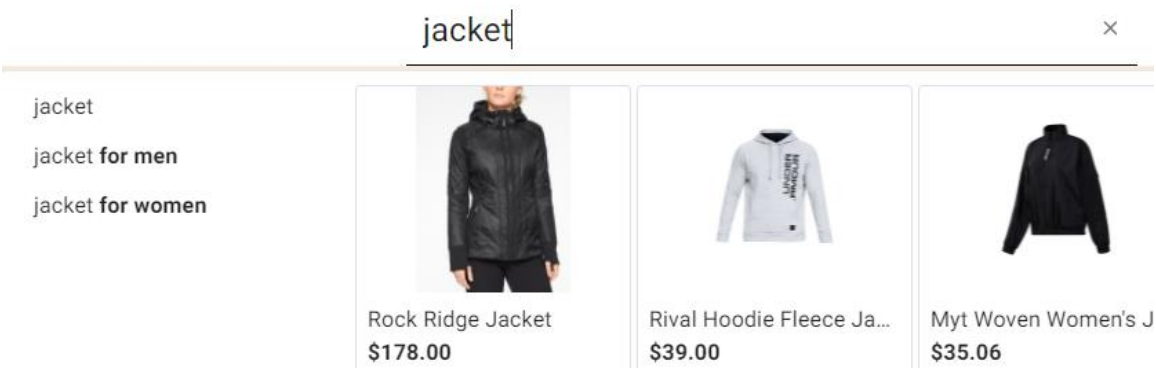
Category suggestions use the query suggestions from Machine Learning and add a category next to it. For example, searching for **jacket** might return **jacket in hiking** and **jacket in winter wear**, two queries that have different intent and should return different results. The categories suggested are taken directly from your catalog and are only shown when pertinent results would be returned if selected.



Product Suggestions

Suggesting products from the search box can be a dangerous feature if not implemented properly. Instead of making a query with the partial keywords the user has entered, it's a better idea to rely on the results from query suggestions to complete the intended query, and then perform a product query based on that suggestion. When doing so, it's always better to still show the textual completions to the query, and potentially only show products after the suggestion is hovered.

You can achieve this with Headless using the **InstantResults** controller.



Creating Listing Pages

Listing pages, sometimes referred to as Product Listing Pages (PLP), benefit from being powered by Coveo, so that the most relevant items from a category are shown first.

When configuring your listing pages, make sure they all share the same search hub. To differentiate between the different pages, you should leverage the tab parameter.

For example, on a site that serves both shirts and jackets, the search hub for both pages should be **Brand Listing**, where Brand is the name of your brand. The shirt listing page should have its tab parameter as **shirt**, and the jacket listing page should have its tab parameter as **jacket**.

Adding Product Recommendations

Product recommendations allow you to recommend products based on the current user context, such as which items they have looked at, what they have purchased in the past, or what they currently have in their cart.

There are multiple **different strategies** you can use with recommendations. You can select which strategy to use when associating a model to a query pipeline.

It is recommended you use the same model for your different strategies. To know which model to query, you should leverage the Recommendation parameter, which you will then be able to leverage in a query pipeline condition.

The Recommendation parameter should contain the location of the recommendation panel. For example, if the panel is the second carousel on the product detail page, you would want to send **Second Product Carousel** in the Recommendation parameter.

If you need help knowing where to put which type of recommendation model, you can get inspiration from the following table.

Recommendation Type	Location
Cart recommender	Cart page
Frequently bought together	Product display page
Frequently viewed together	Product display page
Popular items (bought)	Home page No results page on the search page
Popular items (viewed)	Home page No results page on the search page
User recommender	Home page

You can add the appropriate parameter with a little bit of code. To learn where to add this script, see the **Atomic steps** or the **Headless steps**.

```
preprocessRequest: (request, clientOrigin) => {
 if (clientOrigin === 'searchApiFetch') {
  const body = JSON.parse(request.body);
  body["Recommendation"] = "Your Location"; // Replace this value with the location of your component
  request.body = JSON.stringify(body);
 }
 return request;
}
```

Styling

Styling Coveo components is done through normal CSS. When using the Headless framework, you have full control over the HTML content rendered, including its styling.

The components of the Atomic framework leverage the **Shadow DOM** technology, allowing you to style Atomic components without overwriting the styling of other parts of your website. To learn more on this topic, see **Themes and Visual Customization**.

Tip: The Coveo search interface and the related controls are designed to provide the best-in-class experience to site visitors. Although it's recommended to change the default style to fit your solution, changing its behavior might be a step in the wrong direction for the user experience.

Templating Results

When rendering results, you are encouraged to leverage result templates, available in both Headless and Atomic. Result templates dictate how results are displayed on your page, typically leveraging other components themselves.

You can have multiple templates, which contain conditions, when you want some products to be displayed with a different layout than other templates.

For more information on templates, see Headless **Result Templates** and Atomic **Displaying Results**.

Tracking Clicks

With the Atomic framework, clicks that open the result are automatically tracked. In Headless, clicks need to be tracked based on how you designed your templates.

To track clicks in Headless, you should leverage the **InteractiveResult controller**.

Sending Context Data

In a Coveo Commerce implementation, you normally want to send additional data to Coveo to fully benefit from the commerce capabilities. The expected context values are:

- **website**: the name of the website the user is on. You normally want the name of the website, not the full URL. You also want to make sure the website value contains the country if you have different sites per country. **ACME Canada** would be a perfect example value for the ACME website in Canada.
- **language**: the language the user is in. You want to make sure you send the appropriate **locale value for Machine Learning**.
- **brand**: the name of the brand of the website. Where the website value would contain the country, this parameter would not.
- **name**: the name of the product clicked.
- **id**: the id of the product clicked. Typically, this is the SKU.
- **category**: the category of the item clicked on. For example, if the user has clicked on a laptop, we would expect the click to include the category **Laptop**. Alternatively, if you have a hierarchical structure to your categories, you can send it as **Electronics/Computers/Laptop**, assuming this is how your hierarchy is defined.
- **price**: the price of the item that has been clicked.

Additionally, you may want to send more context, which falls outside of these parameters. This can be done, for example if a logged in user has a profile and you would like Machine Learning to benefit from this information.

To do so in Headless, leverage the **Context controller** of the search engine.

Since the Atomic version of this controller does not currently exist, you will also need to leverage the controller through Headless when using Atomic. For more information on how to do that, see **Access Headless Through Atomic**.

Search Engine Optimization

Having search-powered pages is beneficial for customers who are expecting products adapted to their needs, but less beneficial for search engine crawlers (like Google for example) who expect pages to be static.

There are a few different approaches to ensuring you have good SEO's on your dynamic pages, including using pre-rendered pages or server-side rendering them. For more information and alternatives on this topic, see **External Search Engine Optimization (SEO)**.

Localizing the Interface

By default, the strings from the Atomic search components are in English. You can localize those strings to different languages, or even create your own custom labels, using the built in **i18next** framework. Only English and French are currently available out of the box; you can add your own translations for different languages through i18next.

For more information on localization in Atomic, see **Localization**.

In Headless, there are no out of the box strings, since you are in charge of generating all the HTML rendering.

Improving Relevance

By this point, you should be indexing your catalog and serving the products to the users, while capturing the ways in which they are interacting with your site. The next step is to use that data to improve the relevance of your site.

Machine Learning Models

Coveo offers multiple machine learning models, which you can leverage to enhance the experience on your site.

For a machine learning model to learn from data you are collecting, it simply needs to be created. However, for it to affect your site, you need to make sure it is associated with the appropriate query pipeline.

Coveo separates learnings according to three main criteria: **language**, **search hub**, and **tab**. Those criteria act as hard separation; learning from one search hub will not be shared with another search hub.

You can also send custom contexts to perform "soft" separation of learnings, where the models learn from all events but weigh similar contexts together, as noted earlier.

Automatic Relevance Tuning

Automatic Relevance Tuning (ART) is the machine learning model that reorders the results according to what has been more successful in the past for similar users. With ART, success is based on how often a result is clicked for a given query. Additionally, if you are tracking the purchase events, ART will use it to improve its relevance.

When associating the ART model, you want to make sure you check the **Match the query** option on your ART model; this is so ART does not start injecting results the user is not searching for on your pages. Additionally, you want to make it so it can boost up to 50 results as opposed to the default value of 5, by changing the **maxRecommendations** option. This, combined with Dynamic Navigation Experience, will yield a much more relevant and personalized experience on listing pages.

Query Suggestions

Query Suggestions (QS) is the machine learning model that shows previously searched terms in the search box. Like ART, QS only considers successful queries, i.e., queries that have previously led to clicks.

Predictive Query Suggestions

The **Predictive Query Suggestions** (PQS) machine learning model was designed specifically for Coveo Commerce, and works both from analytics data from your users and from your catalog source.

Not all organizations have access to this model. If you would like to get this model, please **Contact Us**.

Dynamic Navigation Experience

Dynamic Navigation Experience (DNE) is a machine learning model handling facet filters and category boosting. Like ART and QS, DNE relies on search and click events to determine which queries and results are successful.

The main use of DNE is category boosting. By enabling DNE, a user who has previously searched for golf accessories and then starts searching for gloves will be offered golf gloves first. Additionally, DNE boosts popular categories at the top, in complement to ART's boosting of popular individual products.

The second main use of DNE is automatic category selection. With this feature enabled, DNE can automatically select a value from a facet based on the query of the user. For example, someone performing the query "laptop" would automatically have the **Electronics/Computers/Laptop** category selected for them based on their query and on the behavior of previous users.

Product Recommendations

The **Product Recommendation** (PR) machine learning model takes input from the commerce specific analytics events to make links between the different products.

The PR model contains multiple recommendation strategies, which rely on different commerce specific events.

Recommendation queries need to be sent to a dedicated pipeline only used for recommendations. However, you can send all your recommendation queries to the same pipeline.

You are encouraged to use the same model and query pipeline for all your product recommendation strategies. To achieve this, you can associate the model multiple times, but each with a different condition, based on the added Recommendation parameter.

Query Pipelines

Query Pipelines are where rules can be added to modify a query or alter results. This is where Machine Learning can influence results, and where your business rules take effect.

Every query sent to Coveo has to go through a **Query Pipeline**. To determine which pipeline to go through, you are strongly encouraged to leverage **conditions**, specifically ones based on the search hub value.

Query Pipeline Architecture

As a general rule, you want one pipeline per search hub, per language. This is because different search pages with different content tend to require different business rules (e.g., different category boosting or different thesaurus entries).

For more information on the best way to design a query pipeline architecture, see **Designing an Architecture for Many Search Experiences**.

Adding Product Grouping and Variants

It is very common to have related products that should be grouped together on the search page, such as color variations or the low- and high-end versions of a product. To achieve this, you want to leverage product grouping.

Product grouping needs to be enabled in your query pipeline, by adding the **filterField** option. Adding the **filterField** returns related products grouped as a single result, with the most relevant product first, and the related ones as children.

Additionally, to enable variants and availabilities, you need to ensure the catalog is associated with your pipeline. For more information on this topic, see **Leverage Variants and Availabilities**.

Query Pipeline Rules Must-Haves

In a commerce use case, there are a few query pipeline rules that you want to implement. These rules are there to tweak the query to ensure it can return the most relevant results. For more information on those rules, see **Recommended Query Pipeline Configurations**.

Query Pipeline Additional Rules

On top of the rules mentioned above, you can add more rules to help improve the overall search experience. You can see a full list of query pipeline features on the **What's a Query Pipeline** page. Be careful with how you use them; adding too many rules can hinder the overall experience.

If you are uncertain about whether adding your rule will help or hinder the experience, consider using the **A/B Test** feature of query pipelines.

Pitfalls and Misconceptions

When developing a search experience, there are some pitfalls to avoid and misconceptions to debunk. The following examples will help you avoid these pitfalls as you are building your experience.

Search-As-You-Type

Search-as-you-type requires many queries to be efficient. If the threshold is too high, the user will probably go faster than the refresh rate of your interface, and if the threshold is too low, the interface will flicker and move too quickly.

MISCONCEPTIONS	FACTS
Providing a search-as-you-type experience is the standard approach in modern search technology.	The most useful and common experience is called "type-ahead." The suggestions are queries, not results. It's fast, light, and it directs the users towards a successful query.

Using Ranking Tools Without Data

There are several tools to manage ranking. Those tools should be used based on data, such as what is reported in the Coveo Usage Analytics. Using them blindly can have a disastrous effect on the relevance of the solution.

MISCONCEPTIONS	FACTS
I want to find only one type of item, so I will boost that item to a high value.	When applying boosting options, go incrementally to observe every change. If needed, leverage Coveo A/B testing.

Wildcarding

Wildcarding is an option where every word starting with the characters entered in as input will be retrieved. A double wildcard can find the characters entered in any position of any word in the index. Wildcarding can be useful when primarily searching for people, but better results can be achieved with machine learning and query suggestions.

MISCONCEPTIONS	FACTS
It's the only way to get results for a partially entered keyword.	The user won't be entering a partial keyword as they will have query suggestions, which works with the wildcard concept, but in a much more efficient way.
Users won't find what they're looking for without wildcarding.	When using wildcarding, more results are returned, and most of them will be out of context. The point here isn't to find more items, but to find the relevant one!

Implementation Checklist

Analytics and Machine Learning

- [ ] Are you tracking the standard Coveo events from your search components?
- [ ] Are you tracking commerce specific events (detail, purchase, add, etc.)?
- [ ] Did you create your Machine Learning models (ART, QS/PQS, DNE, Recommendations)?

Index

- [ ] Are all your products, variants, and availabilities indexed in your Catalog source?
- [ ] Are your products, variants, and availabilities connected through your Catalog?
- [ ] Are you using the standard fields for metadata?
- [ ] Do you have a system in place to easily update content without pushing the whole catalog again?

Query Pipelines

- [ ] Did you organize your query pipelines in a relevant manner?
- [ ] Did you ensure recommendations go through their dedicated pipeline?
- [ ] Is your catalog associated with the appropriate query pipelines?
- [ ] Are your Machine Learning models associated with your pipelines?

Search Experience

- [ ] Are you leveraging search hubs with appropriate names?
- [ ] Does your global search box share the same search hub value as the page it redirects to?
- [ ] Do your listing pages share the same search hub, but have different tab values?
- [ ] Are you leveraging Product Recommendations?
- [ ] Did you configure your listing pages to be SEO friendly?

Coveo Resources

Coveo Docs

Coveo Docs is the official documentation for all Coveo products and features. Use the search bar at the top of the page to find the topics you need when developing with Coveo.

Coveo Level Up

Coveo Level Up is the education portal of Coveo. Register for learning events, complete training courses and tutorials, and compete with other people at your company to be at the top of the leaderboard.

Coveo Connect

Coveo Connect provides access to the Coveo community, where you can ask questions to the community on the forum to get answers to your questions. It is also where you can open tickets with Coveo Support.