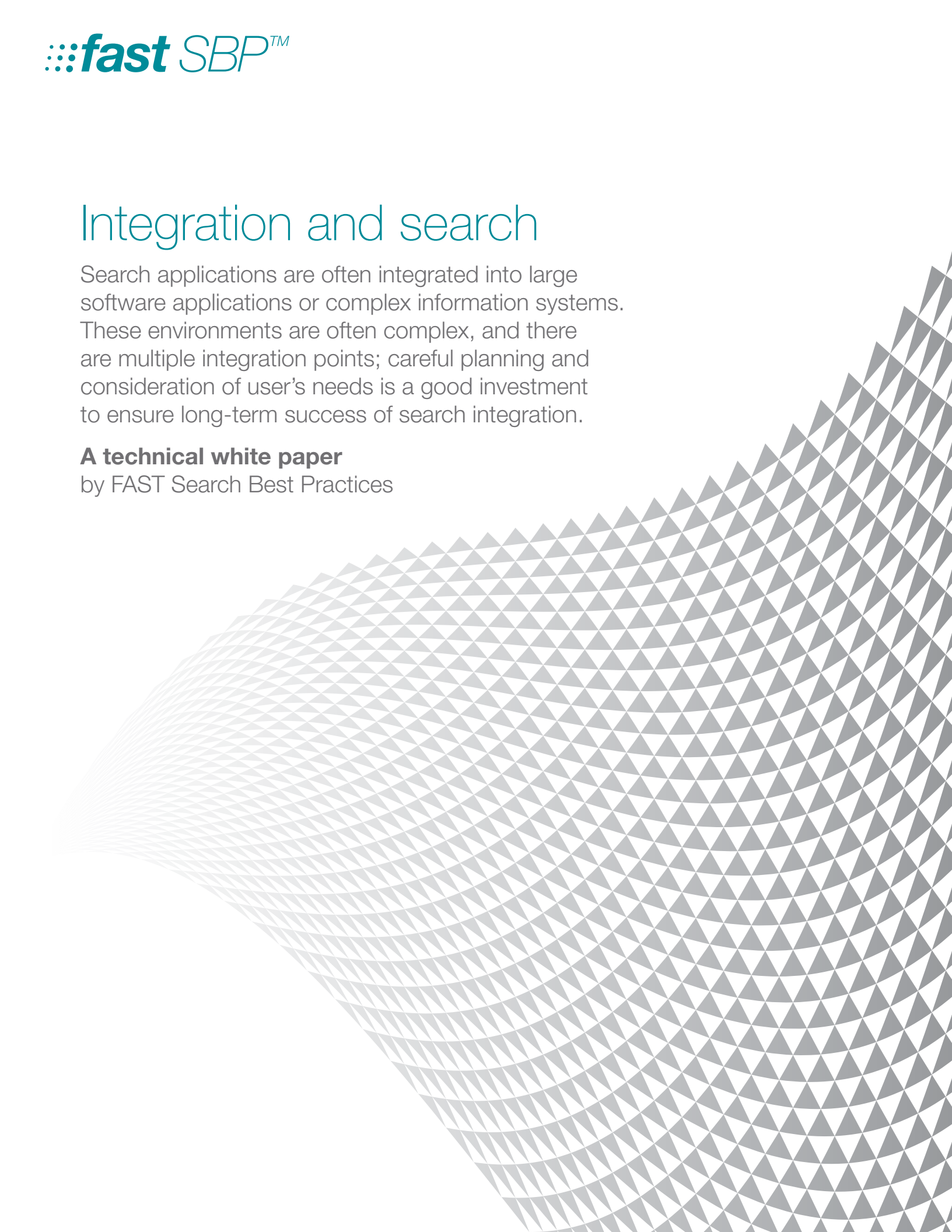
fast SBP™

Integration and search

Search applications are often integrated into large software applications or complex information systems. These environments are often complex, and there are multiple integration points; careful planning and consideration of user's needs is a good investment to ensure long-term success of search integration.

A technical white paper

by FAST Search Best Practices



5 things you should know about integration

1. Integration is the act of embedding third-party software into an application
2. Developing a powerful search engine from scratch is a long and complex process
3. Search engine integration is done with a modular approach, using flexible APIs for content indexing, querying, and administration
4. Content push provides flexibility and the ability to integrate advanced error handling
5. A well-integrated search engine can power a lot more than just search

Why is integration important for good search?

These days, users take it for granted that every Web site or computer application contains a search function. It may simply be a character string matching feature, such as that commonly included in text editors, or it might be a database-driven field lookup – for example, finding contact addresses from exact names in a Customer Relationship Management (CRM) application.

Simple string matching is no longer satisfactory for most users, given the rise in their familiarity with search and the increase in variable data types (such as Microsoft Office documents, pdfs, Web pages, and e-mail). They expect the full power of search, including fast query response times, dynamic and highlighted teasers, drill-down navigators, advanced linguistic optimizations, searching across many fields at once, and Boolean operators.

That helps explain why many new software products now include search features, and why existing products are upgraded to include advanced retrieval capabilities. The decision for application publishers is whether to task an internal team with developing the search algorithms and the appropriate software in-house or whether they should evaluate off-the-shelf search engines and their APIs (Application Programming Interfaces) and select the one with the required features and the most appropriate and flexible integration tools.

Two areas where integration is typical are an authoring or management application such as a Document Management System (DMS), where stored content has to be searchable, and an industry-specific workflow and investigation tool (e.g., for law firms or for compliance in the financial services sector) where many external data sources are processed.

If the preferred option is to license software, it is essential to make sure that the integration is carried out effectively. A successful integration will translate to a seamlessly improved search experience within a familiar and well-liked product without compromising the product's security, scalability, or other key features.

To build or to buy

Most companies with in-house development teams will sooner or later face the buy-versus-build decision. They have to take into account the programming language of the existing application, how deep or how comprehensive the integration needs to be, and how much the existing technology will have to be customized.

The advantages of developing a proprietary in-house solution include the complete control of development cycles and commercial flexibility of pricing. The assessment will also weigh the financial consideration of the time required to build and maintain an internal solution versus the license and royalties specified in an OEM (original equipment manufacturer) agreement. So it is important to bear in mind the cost of developing the core engine, as well as the cost of the administrative and configuration utilities, scaling and failover mechanics, documentation, etc.

Since the functionality requirements of search are increasingly costly, the investment required to create an offering that can compete with off-the-shelf solutions has never been higher.

The primary advantages of an OEM search solution are its ability to provide world-class search while minimizing the financial outlay and the associated execution risks and reducing the time-to-market by leveraging tried and tested modules. In general, software vendors will select an OEM solution so they can eliminate development costs and avoid the potential difficulties if users are offering substandard application-centric search capabilities.

Integration points

Despite being off-the-shelf, OEM integration will require substantial planning to provide a seamless assimilation of the two technologies. The component architecture of search engines needs to be understood so that each connection point is identified and treated separately.

There are five main areas to be considered: content creation for indexing, index configuration, query logic, user interface design, and administration and configuration. Below is a quick review of each:

1 - Content aggregation: this refers to the process of feeding data to the search engine's ingestion process to create the index.

The first way to feed content into the ingestion process is to use an off-the-shelf connector. Simple connectors are a file system traverser, which monitors directories for new, modified, and deleted documents, a Web crawler which does the same for Web pages, or a database connector which will use Simple Query Language (SQL) to extract structured data and embedded documents. Integrators can also leverage connectors dedicated to repositories, such as Lotus Notes or Documentum. These may be CRM applications, email systems or legacy data stores. Best practice is for the connectors to be preconfigured by the OEM, based on prior knowledge of repository installation types found at customer sites. Alternatively, a full installation of the connector will need to be performed by the end client, the OEM's professional services team, or by an implementation partner , although this can increase the installation and configuration complexity of the application.

Using off-the-shelf connectors allows technology partners to rapidly support many repository types. The other option is for connectors to be developed using the search engine's indexing API (described below). This gives the flexibility to support data sources that may be particular to an industry, or where no standard connectors are available.

A closely coupled content-side integration leverages a content indexing API to push data to the search engine. When using such an approach, all connections to the original data store are made by the calling application, which then has complete control over scheduling, interfacing protocols, and data structures. The API also supports error-logging callbacks.

Taking advantage of these callbacks, actions may be triggered when documents fail to be processed or do not make it into the index. For example, for auditing purposes a compliance or storage application will keep a report of all documents that did not get indexed.

The indexing integration should include provision for updates and deletions if the system has dynamic content. This will be handled by an off-the-shelf connector if used, but must be built in separately when an API document push method is chosen.

Whatever the method chosen, the data is then pushed to the document-processing stage which accepts either text-based (HTML, XML) or binary formats (e.g., Microsoft Office, pdf, or multimedia files), along with the metadata associated with these documents.

2 - Index configuration refers to the configuration and tuning of the search engine. The best search engines provide a complex and highly tunable search experience. For example, they offer the ability to weight different fields within a document when searching. Different fields can also be configured for other purposes: querying, sorting, navigators, and range restrictions. Nonetheless, while a search engine is designed to be very flexible (in terms of metadata schemas etc), a search-enabled application will have a common (fixed) implementation structure. Therefore index configuration decisions should be made upfront by the application designer rather than by allowing end users to choose.

Another key element when designing an index is security. This includes document and attribute level security. Either the application performs a standard search and then filters out the documents that the search user does not have permission to see, or security Access Control Lists (ACLs) are indexed as metadata to allow filtering by the core search engine. The chapter on "Security" details this, as well as all the other aspects of a secure search implementation such as communications encryption and user authentication.

The final crucial aspect of configuration is the need to understand benchmarking and the installation footprint. The OEM must consider what type of application profile is being supported – high-volume, high QPS (queries per second), or both – and also understand the key constraints. Is search a significant part of the solution, and will the end clients be prepared to dedicate the appropriate hardware resources to the search

application? Or is search just a "nice to have" feature that should have a minimal effect on the application's installation and hardware footprint?

3 - Query logic and results processing: this addresses the processing of queries and post-processing of results.

Query logic concerns the query syntax that is exposed to the end user or that is leveraged by the application behind the scenes, and any transformations between the two. It also relates to relevancy configuration, such as whether to weight the freshness of a document in the ranking of hits and any business-specific logic such as synonyms or query disambiguation.

Q: My application is intended to run on mobile phones. Should I use a thin or thick client?

A: With mobile phone applications, where bandwidth is at a premium and the possibilities with WML (Wireless Markup Language) are limited, a Java-based client installed on the phone may enable a more intuitive and fast experience. Advanced XML indexing and searching capabilities would allow querying across it.

Regarding the processing of results, some OEMs choose to push into the index only the text that will be searched or filtered, and not the metadata that is used for display purposes. One example: the quantity of items in stock for an e-commerce site.

When display information stored in the native application is required, the results must be post-processed, with additional API calls to the source performed. In cases where everything required in the results set is stored in the search engine, the search hit list can be populated directly, with the appropriate formatting applied.

It is important to understand that if one particular field has a higher modification frequency when compared to the rest of the document, it may not be advisable to store the information in the search engine.

4 - User Interface design: this covers how both the query interface and the results presentation are built into the application.

From a UI viewpoint, the planning of a search OEM is very similar to classic search design. Typical decisions involve whether or not to embed search within an existing page or screen, how to apply navigators for browsing in the result set, and whether to include advanced search options, such as metadata drill-downs or filter boxes.

When designing a UI a decision must be made about the nature of the client application. One option is to use a minimal thin Web-based client which displays data directly from the search engine. In this scenario, the results are manipulated only via templating. More sophisticated thin clients will parse the hit list returned from the search engine in XML, to then process and format the results. This is typically done in a JSP or a similar server-side technology. A thick client is the third option, where a program running on the client computer (for example, as part of the OEM's existing application) processes the results.

5 - Administration and configuration: these are the administrative and configuration UIs and APIs (the extent of which will vary from one technology to another). A key goal is deciding how much of the search engine's configuration and administration is left for the end clients of the application, versus how much is pre-defined when the product is developed.

For features deemed necessary for the client to tweak, the tools to do so must be built into an existing administrative UI, using the search engine's administrative API. In general, though, giving direct access to the search engine's administrative UI is bad practice because it will expose too much to the client, creating confusion, and permitting more modifications than may be appropriate. It will also make each installation more varied and therefore trickier to support.

Complex integrations and best practices

The decision to buy versus build will be based on the need to leverage tried and tested technology, and to keep the company's focus on its core competencies. Therefore, the first key recommendation when embedding search within another application is to investigate the capabilities and APIs provided, to identify which parts are reusable. Here is a quick look at three of those facets:

Content creation

There are two principal types of search OEM. One has applications that deal with data storage, such as a document management, records management, or archiving solution. The other provides a bridge across one or many sources, such as a compliance application (spanning e-mail and a DMS) or a litigation support tool (for example, indexing everything on confiscated hard drives in order to find valuable evidence for use in trials.)

When there is only one repository of data to be considered, content should be aggregated using a file system share or database, or by feeding documents to the content API.

In that example, integration requirements should be minimal. For file systems, a shared folder to which both applications have access is used as a landing area for new documents. The search engine's file traverser monitors and indexes any new documents within that folder. Or, if the native application uses a file system for the internal storage of content, the traverser can be given direct access to the appropriate directories. Mechanisms to add metadata with this simple method are available. Similarly for the database approach: a temporary table can be created with the relevant information (required metadata, path to the binary file, etc.), the connector can be run against the original tables, or a view thereof.

In the second option, the content API has the advantage of error callbacks to track the document throughout its indexing lifecycle. Content pushing allows the OEM to control the indexing flow and lag between document creation and indexing much more tightly. This is critical when each document must be accounted for, or when an action such as a re-try, or an alert is triggered for failed documents.

Using the API approach, both binary data and metadata are pushed together, They are handled by the document processing part of the search engine for conversion into plain text and manipulation. Ready to index text can be submitted in XML format, for example. XML fields can contain paths to documents, enabling hierarchical models with binary data to be indexed. A sophisticated OEM integration may also include complex data structures, such as hierarchical document models, requiring the use of XML for indexing.

Q: My organization is running a sophisticated wiki application. Can a search engine support a suitable document model?

A: The assumption is that your wiki application contains many entries, each with many comments and updates. These would all need to be indexed as nested entries with their own metadata (such as author and edit date) to allow searching within the scope of a single edit or the whole wiki entry. It's possible to use an XML schema that would support this and a search engine with advanced XML indexing and searching capabilities would allow querying across it.

Enriching and modifying data can increase the effectiveness of queries. For example, if the application is targeted at law firms and indexes attorney memos and other legal documents, a custom entity extraction module could be used to tag all legal citations for cross-referencing and navigation. When field manipulation is required, the data can either be pre-treated by the source application before ingestion, or the framework and tools from the search engine's document processing stage can be used.
Data processing is also used to reduce an index's size. If the application is an archiving solution, it may be sufficient to index key metadata, or extracted top terms, rather than the whole text of each document. The removal of duplicate terms, or lemmatisation by reduction (each word is reduced to its base form) will also reduce the size of the index.

Index configuration

Search engines are used for a range of applications, with a variety of data and query requirements. In a backup and archiving application, for example, large volumes of data will need to be indexed, and disk and memory usage should be kept to a minimum. An online application searching a large repository, such as a set of scientific or legal documents, will be concerned with scaling in number of documents, in addition to supporting enhanced functionality and a higher rate of QPS. On the other hand, a niche content owner, such as a provider of mobile-phone ringtones, will have a small volume of data but will need to maintain a high QPS. In this scenario, index latency will be unacceptable because it will almost certainly lead to missed sales.

This is one of the main areas where the OEM can reduce the configuration complexity of search that is exposed to its customers: by configuring memory settings (capping which parts of the search engine use the most memory), or tuning options that increase or decrease the disk footprint (different field processing and types, such as integer fields, parametric navigators, etc.). Based on an understanding of typical usage, OEMs can configure an application once so it satisfies the needs of most customers.

Query and results processing

One of the most common query transformations required for OEMs is when the application has an existing search feature – for example, home-grown or from another search vendor – and the users are already familiar with the syntax.

More often than not, the new search engine will be able to support the same types of logic as the previous one (e.g., Boolean, nested, proximity queries, etc.) but the rules and syntax will be different. Therefore, care must be taken when writing the translation element.

Another common OEM request stems from the provider having its own source of content over and above the customer's managed content. An example is a news publishing CMS: the technology provider may have a source of news feeds which the client can access as part of the licence, or as an upsell option. The recommended solution is to centrally index the common data (the news feeds), using the same search engine technology, albeit configured to support greater query loads. When performing a search, the application installed at each client will fire off two queries. One is against the local search engine, which has local content indexed and the remote installation in the data center, merging the results locally within the search broker layer. This will minimize the content and installation that is managed at the customer's data center.

Operational integration alignment

Commercial software packages will have best practices of their own, including high-availability strategies, usage patterns, and back-up policies. A successfully embedded search engine will be installed and configured to be in sync with these application-specific policies. Only under these circumstances will the combined package of the original application and search be able to scale seamlessly and ensure a uniform level of service. For example, a DMS that scales to one billion documents will need a search engine that can do the same; if the search tool cannot, it will be obliged to offer an incomplete solution. Ideally, the coupling will have be executed in such a way that when scaling or backing-up the core data modules, search will automatically and seamlessly follow.

Another area of alignment is operating system and language support. For whatever the application already supports, the search engine must naturally be configured to suit.

Mini case study
Storage provider adds search and chargeback to product line

Who
A Leading worldwide provider of enterprise storage solutions

Challenge
To index large data volumes (more than 50 million documents per server) with high- availability configurations, integrated administration, and security.

Solution
Stored data is tagged with hierarchical XML in varying schemas containing document metadata. This XML is pushed into the search engine using the content API. The charging and reporting tool gives administrators a unique birds-eye view of the usage of storage within the enterprise, such as data size usage per department, using analytics on navigator fields.

Maximising the search ROI

Good search tools include other features and functions on top of their basic search capabilities. Alerting, browsing by metadata, data storage, "latest news" pages, expert locators, and collaborative filtering are among many possible features that can be added to traditional search. None of those features uses search in the traditional sense of a box where a user enters a keyword. However, once data is indexed into the search engine, these are quick-win additions that can be used to augment a product. For example:

Alerting: the integration effort will have already developed a mechanism pushing each published document through a conversion filter to extract the text. Most search engines provide a module to match this text against predefined rules to trigger alerts to users.

Browsing: if the search engine has indexed drill-down navigator information, this is used to refine hit lists. These drill-downs can be used for a document browsing feature (e.g.; browsing the content in a DMS by author, folder, or file type) which would be powered by parametric searches.

Data storage: if size of application is a concern, one solution is to add search without affecting the installation footprint. This is achieved by replacing the existing text data storage with the search engine itself. That is, whereas previously the text data was stored in a file system or a database and the user interface displayed contents by sending requests to that store, now all data display actions will be queries to the search engine, which holds a unique version of each document.

More features can be used by the OEM to increase the product's appeal, or sold as functional add-ons to grow licensing potential without substantial development work.OEMs can further increase revenue generation with traditional search SI (Systems Integrator) work. As discussed, a key objective is to hide the complexity of search from the customer. Nonetheless, the tuning of relevancy models, the adding of new connectors and data sources, the development of industry- and customer-specific taxonomies or entity dictionaries, and GUI design can all be positive offerings for customers looking for a more complete search experience.

Fundamental steps to seamless integration

The most important part of integration is deciding which configuration or query features to expose to the consumers and managers. Quality enterprise search engines are very flexible tools. They are designed to cope with many different types of applications and industries: e-commerce, knowledge management, archiving, video search, etc. An OEM is typically targeted at a subset of the areas where search engines must compete – for example, a niche application within one industry. Therefore, for each decision about design and configuration made by someone installing a search engine (what relevancy model to use, which fields to make searchable, how to tune for optimal indexing and querying speed), the OEM must decide whether the decision can be generalized to all its clients or whether the option must be left open for systems integrators or IT administrators to fine-tune the system.

This is a crucial point; after all, the first line of support when something fails is the integrator, not the search engine vendor. By reducing the number of installation permutations, the search engine will become easier to trouble-shoot when customers flag problems.

OEMs should be aware of the potential revenue opportunities arising from search - for instance, quick-win add-ons and potential SI-style customization work. These financial gains on top of the increased competitiveness of any product that can perform world-class search, justify the thoughtful planning that must go into ensuring seamless integration of search.

In addition to the above, common best-practice recommendations for OEMs are to:

- Use a content push method to enable flexibility and error-checking within the indexing side
- Modify advanced index settings to fine-tune the footprint of the index dependent on the impact that search has on the application
- Evaluate the pros and cons of using the search engine as a possible store for the data as well as an index
- Favor a loose integration to begin with if turn-around times are short
- Consider a deeper assimilation of the technologies later to increase the benefits the partnership.

Frequently asked questions

Q: What is integration?

A:Integration is the joining of two or more software systems at the logical or physical levels.

Q: What is an API?

A: An Application Programming Interface (API) is a software interface that enables developers to access the features and functions of a hardware or software platform.

Q: What is the difference between a content API and a query API in the context of search engines?

A: A content- or document-importing API is designed to feed documents and metadata into a search engine. A query API is used to execute queries and obtain a hit list in a usable format.

About FAST SBP™ (Search Best Practices)

SBP consulting is a highly focused transfer of search knowledge and experience from FAST to its prospects and customers. SBP workshops aim to help enterprises realize the full potential of search, by creating optimal strategic, functional and technical roadmaps, delivered in the form of business model, solution and architecture designs.

Fast Search & Transfer
www.fastsearch.com
info@fastsearch.com

Regional Headquarters

The Americas
+1 781 304 2400

Europe, Middle East & Africa (EMEA)
+47 23 01 12 00

Japan
+81 3 5511 4343

Asia Pacific
+612 9929 7725

SWP.005.T.01.011206