Level 3: Advanced | Digital Librarian

Module 17

AI & library systems integration

For digital librarians ready to connect AI to the systems they manage (ILS, repositories, discovery layers) without a computer science degree. This module demystifies APIs and shows you what's actually possible.

By Yulia Brusova

15 min read

WHAT YOU'LL BE ABLE TO DO

1. Explain what an API is and how it enables AI integration with library systems
2. Identify integration opportunities between your ILS and AI tools
3. Evaluate whether to build, configure, or advocate for a vendor feature
4. Understand the data pipeline concept in the context of library systems
5. Read basic API documentation well enough to evaluate integration possibilities
6. Conduct a vendor AI due diligence review covering data privacy, algorithmic transparency, environmental cost, and contract terms, using ALA's draft AI guidance as a framework

> *The most common question among digital librarians about AI integration is whether a given capability is something a practitioner can implement independently or whether it requires a developer. The honest answer is that it depends on the specific integration, the library systems involved, and the institution's technical environment. Some AI integrations are accessible today without any programming knowledge, some are accessible with modest technical assistance, and some genuinely require developer involvement or a vendor solution. This module provides the grounding needed to know which category a given integration falls into, and to have informed conversations with IT departments and vendors accordingly.*

What an API is and why it matters for AI integration

An API, or Application Programming Interface, is a defined channel through which two software systems exchange data. The technical acronym is less important than the practical concept: when one system needs to send information to another, or request information from another, an API is the standardized mechanism that makes that exchange possible without requiring the two systems to be built by the same people or to understand each other's internal architecture.

For example, when a patron searches the library's discovery layer and receives results simultaneously from multiple licensed databases, APIs are what make that coordination possible. Such a search involves the discovery layer sending requests to each database's API, receiving responses in a standardized format, and presenting the combined results as a

single interface. Similarly, when a library website displays current hours pulled automatically from an ILS rather than maintained manually, an API is typically involved in that connection.

The reason this matters for AI integration is that AI tools increasingly expose APIs of their own. For example, Claude, ChatGPT, and most other major AI platforms offer APIs that allow other systems to send them content and receive processed responses automatically. Such a connection means that instead of a library staff member manually copying catalog records into an AI interface and pasting the output back into another system, the transfer can occur programmatically, at scale, and on a schedule. The ILS can send records to an AI service for description enhancement, the discovery layer can request AI-generated summaries, and the chat reference tool can use an AI API to draft initial responses, all without manual intervention at each step.

It is evident that understanding APIs at a conceptual level, even without the ability to build API connections independently, changes how a digital librarian evaluates vendor products and frames conversations with IT departments. The practitioner who understands that an AI integration is technically possible through existing API infrastructure is in a fundamentally different position in those conversations than one who does not.

What AI integration looks like in major ILS platforms

The practical possibilities for AI integration depend substantially on which integrated library system a library operates. There is no single answer that applies across all platforms, and the state of vendor-built AI features is changing rapidly enough that consulting current documentation and speaking directly with vendors is essential before drawing conclusions about what is or is not possible.

Alma, developed by Ex Libris, has one of the most accessible API ecosystems among commercial ILS platforms and an active developer community. Ex Libris also offers LibOW, a low-code automation platform that connects Alma to external services including AI tools without requiring custom code. In order to evaluate what AI integrations are possible for an Alma installation, the practitioner should examine LibOW's available connectors before commissioning any custom development, as the functionality required may already be available through the platform's built-in tools. For example, automated metadata enhancement on record import, batch processing of records through an external AI service, and reporting workflows fed from Alma data are all practically achievable through the Alma API ecosystem with varying degrees of technical support required.

Sierra and other Innovative platforms use an older API architecture that is functional but typically requires developer involvement for custom integrations. Such platforms are not precluded from AI integration, but the practical path more commonly runs through vendor-built features than through custom development. Koha, as open-source software, offers greater flexibility for custom API integration than most commercial platforms, provided the institution has technical staff or community support available to implement the connection.

There is no doubt that the most important first step before investigating any custom integration is a direct conversation with the ILS vendor. Vendors across the ILS market are actively building AI features into their platforms, and the question of what AI integrations a library needs may already be on the vendor's roadmap for an upcoming release. Such a conversation takes fifteen minutes and may save months of custom development work. The practitioner who builds a custom metadata enhancement workflow in January may discover in March that the vendor has shipped the same capability as a supported feature. Asking the vendor first is not a sign of limited ambition; it is the operationally sound starting point.

Institutional repositories and AI: accessible entry points

Institutional repository platforms present more accessible AI integration opportunities than integrated library systems for most digital librarians, for two reasons. First, IR platforms are often structurally simpler than ILS systems, with clearer data models and more straightforward API documentation. Second, the AI use cases for repositories are particularly well-defined: metadata enhancement, description generation for underdescribed items, subject term suggestion, and discoverability improvement are all tasks where AI assistance produces clear and measurable value.

For DSpace installations, an active community of practitioners and developers is building AI integrations for metadata suggestion and quality control. For example, workflows that send item records to an AI service and receive suggested subject headings, enhanced abstracts, or normalized title fields in return are in active development and use in the DSpace community. Such developments are best tracked through the DSpace community forums and annual user group meetings rather than through general search, as they move quickly and general sources often lag behind current practice.

For bepress and Digital Commons installations, which are vendor-managed platforms with limited custom integration capability, the practical path is advocacy with the vendor rather than independent development. Such platforms restrict what users can implement outside the vendor's supported feature set, which makes the vendor conversation both more important and more constrained. Ex Libris, which acquired bepress in 2017, has been

integrating AI features across its product line, and the trajectory for Digital Commons is worth monitoring directly.

It is evident that for most IR administrators today, the most practical AI-assisted metadata workflow does not require any API integration at all. For example, a practitioner can export a batch of item records as a CSV file, submit batches to Claude or another AI tool with a structured prompt requesting description enhancement, subject term suggestion, or abstract normalization, and import the reviewed results back into the repository. Such a workflow is manual rather than automated, but it is accessible without technical support and produces meaningful improvements to record quality in a fraction of the time required to work through each record individually. Furthermore, it serves as a proof of concept that can be used to justify a more automated integration when the case for developer time or vendor investment needs to be made.

Data pipelines: what they are and when they are appropriate

A data pipeline is an automated process that extracts data from one system, applies a transformation or processing step, and loads the result into another system, on a recurring schedule and without manual intervention at each step. The concept is not specific to AI, but AI services fit naturally into pipeline architectures because they can serve as the processing step between extraction and loading.

For example, a library data pipeline for AI-assisted metadata enhancement might pull new catalog records nightly from the ILS, submit each record to an AI service with a prompt requesting subject term suggestions and an enhanced description, receive the enriched records back, and load them into a review queue for cataloger approval before the final records are written back to the ILS. Such a pipeline runs automatically on its configured schedule, and the cataloger's involvement is limited to the review step rather than the initial processing. Additionally, the same pipeline structure could be applied to repository records, chat reference logs requiring categorization, or usage statistics requiring pattern analysis.

There is no doubt that data pipelines, as described here, are not independent projects for most digital librarians without developer support. They require API access to both the source and destination systems, server infrastructure to run the pipeline process, error handling to manage failures without data loss, and monitoring to catch problems before they accumulate. In order to implement a true pipeline, the practitioner needs either developer support, a managed integration platform such as Zapier or Make for simpler cases, or a vendor product that includes the pipeline functionality as a supported feature.

Understanding what a pipeline is and what it should do, however, is itself valuable even without the ability to build one. For example, a digital librarian who can describe a desired pipeline in concrete terms - what data should move, from where, to where, on what schedule, with what transformation in between - is far better positioned to advocate for that capability with IT or a vendor than one who can only express a general desire for "better AI integration." Such specificity is what converts a vague request into a scoped project that a developer or vendor can evaluate and implement. The practitioner's role in pipeline development is often less about building and more about defining what the pipeline should accomplish with enough precision that someone else can build it.

Build, configure, or ask the vendor: a decision framework

Every AI integration decision for library systems involves a choice among three paths: asking a vendor to provide the capability as a supported feature, configuring an existing tool to connect systems without custom code, or building a custom integration with developer support. Each path is appropriate under different conditions, and the most common error is choosing to build when advocating to the vendor would have been more effective.

Asking the vendor is the correct first step when the library operates a commercial platform and the use case is general enough that other libraries likely share it. For example, a metadata enhancement feature that thirty libraries would benefit from is something a vendor has a strong incentive to build; one library building it as a custom integration is a poor allocation of effort relative to a well-articulated feature request submitted through the vendor's formal channel. Such requests are most effective when they include a specific description of the workflow, the data involved, and the measurable outcome the library expects. Additionally, advocacy through library consortia and professional organizations amplifies the signal that a feature is broadly needed.

Configuring an existing tool is the appropriate path when a no-code or low-code platform can connect the relevant systems without custom development. For example, Zapier, Make, and Alma's LibOW platform can connect library systems to AI services for a range of workflows without writing any code. Such tools have meaningful limitations - they handle straightforward trigger-action logic more reliably than complex conditional workflows - but for the category of integration they support, they are faster to implement, easier to maintain, and more accessible to practitioners without programming backgrounds than custom development.

Building a custom integration with developer support is appropriate when the use case is specific to the institution's workflow, no vendor product addresses it, and the benefit of the

integration justifies the ongoing maintenance cost. Such a decision requires honest assessment of the maintenance burden, because custom code requires ongoing attention as the systems it connects are updated, and a library that builds a custom integration without a plan for maintaining it is incurring a technical debt that may eventually cost more than the integration saves. The practitioner's responsibility in a build decision is not only to make the case for the initial development but also to ensure that the institutional commitment to maintenance is explicit before work begins.

Vendor AI due diligence: questions worth asking before any contract is signed

Every path described in this module, asking the vendor, configuring an existing tool, or building a custom integration, eventually runs into the same gate: a contract, a license amendment, or a terms-of-service agreement that governs what the vendor's AI feature actually does with the library's data. ALA's draft "Guidance on the Use of Artificial Intelligence in Libraries," released in April 2026 and pending ALA Council action in June 2026, takes up vendor transparency as one of its central themes, and the four areas below, data privacy, algorithmic transparency, environmental cost, and contract terms, are the questions that theme translates into for a digital librarian evaluating a specific AI feature. For the digital librarian evaluating an AI integration, this means due diligence is not a separate compliance step performed by someone else after the technical decision is made; it is part of the same evaluation that determines whether build, configure, or ask-the-vendor is the right path.

Data privacy is the most immediate due diligence question, and the most concrete. When an AI feature processes patron queries, circulation data, or institutional repository content, the practitioner needs to know whether that data leaves the library's systems at all, and if it does, whether it is retained by the vendor, used to train models that serve other customers, or processed by a subcontracted AI provider the library's contract with the ILS vendor may not name directly. For example, a discovery layer's AI summary feature may call an underlying model from a third party AI provider, in which case the library's data is governed by two sets of terms, the ILS vendor's and the AI provider's, and a due diligence review that examines only the first has not actually answered the question.

Algorithmic transparency, sometimes described as the "black box" problem, is the second area, and it is harder to resolve because vendors often cannot, or will not, fully explain how a proprietary AI feature reaches its outputs. What a due diligence review can reasonably ask for is not the model's internal workings but its behavior under specified conditions: what

categories of records or queries does the feature perform poorly on, what happens when it is uncertain, and is there a way to disable or override the feature for a specific collection or workflow if its outputs prove unreliable for that context. A vendor that can answer these behavioral questions, even without disclosing proprietary model details, is offering meaningfully more transparency than one that responds only with marketing language about the feature's capabilities.

Environmental cost is a newer due diligence category, and one increasingly worth raising directly with a vendor. Training and running large AI models consumes substantial energy and water for data center cooling, and a library's procurement decisions are, in aggregate, part of the demand that shapes how that infrastructure is built and operated. A due diligence review does not need to resolve this question definitively, but asking a vendor whether they publish information about the energy sourcing or efficiency of the infrastructure running their AI features is now a legitimate part of the conversation, and a vendor's response, or lack of one, is itself informative.

Contract and licensing questions tie the other three categories together in practical terms. In order to evaluate a vendor's AI feature responsibly, the digital librarian should confirm what the contract says about data ownership and retention after the contract ends, whether the vendor can change the AI feature's behavior or the underlying model without notice, whether the library can opt out of an AI feature for specific collections while retaining the rest of the platform, and who bears responsibility if an AI generated output, such as a suggested subject heading or a generated description, causes a problem after it has been published. Such questions are appropriately directed to the library's procurement or contracts office working alongside the digital librarian, but the digital librarian is often the only person in that conversation who understands what the AI feature actually does, which makes the practitioner's participation in contract review a genuine professional contribution rather than a formality. None of these four categories, privacy, transparency, environmental cost, or contract terms, can be fully resolved by a vendor's answers alone; the final judgment about whether a given AI feature's risk profile is acceptable for a specific library's patrons and collections remains the library's own, made with the vendor's answers as input rather than as a substitute for the decision.

PRACTITIONER NOTE

The most instructive pattern in library systems integration work is building a custom solution for weeks before asking the vendor directly. The vendor's answer, when finally asked, is sometimes that the feature is already on the roadmap for the next release, or

that a supported connector in an automation platform already handles exactly the workflow in question. That conversation takes fifteen minutes. Asking the vendor before building is not a failure of ambition; it is the operationally correct starting point, and the time saved when the vendor says yes can be redirected to the integrations only the institution can build for itself.

KEY TAKEAWAYS

- An API is a standardized channel through which two software systems exchange data; AI tools increasingly expose APIs that allow library systems to send content for processing and receive results automatically.
- What AI integration is possible depends heavily on which ILS or repository platform the library operates; Alma and open-source platforms offer more accessible API ecosystems than most commercial systems.
- For institutional repositories, an AI-assisted metadata workflow using batch CSV export and import is accessible today without any API integration and serves as a practical entry point and a proof of concept for more automated approaches.
- A data pipeline automates the movement and transformation of data between systems on a recurring schedule; building one requires developer support, but defining what a pipeline should do with precision is itself a valuable practitioner contribution.
- The three integration paths are vendor advocacy, configuration of existing tools, and custom development; asking the vendor first is the correct default, and building is appropriate only when the use case is institution-specific and the maintenance commitment is explicit.
- The practitioner's most important contribution to AI integration work is often not building but defining: describing what data should move, between which systems, with what transformation, precisely enough that a developer or vendor can evaluate and implement it.
- ALA's draft Guidance on the Use of Artificial Intelligence in Libraries (April 2026, pending June 2026 Council action) frames vendor transparency as a baseline expectation; a vendor AI due diligence review should cover data privacy, algorithmic transparency, environmental cost, and contract terms, with the digital librarian's technical understanding of the feature itself as essential input to that review.

References

APA 7TH EDITION

1. Ex Libris. (2025). *Alma library services platform.* Clarivate. https://exlibrisgroup.com/products/alma-library-services-platform/
2. Ex Libris. (2025). *LibOW: Low-code automation for Alma.* Clarivate. https://exlibrisgroup.com/blog/what-is-libow/

ACRL AI Competencies covered

Use & Application

Analysis & Evaluation

Sub-competencies: 4.1, 4.4, 3.2 · ACRL AI Competencies (2025)