

Generative AI and Automated Workflow for Knowledge-Centered Support (KCS)

Michael Iantosca

*Senior Director of Knowledge Platforms and Engineering
Avalara*

Enterprise-grade software and products are inherently complex. While the instinctive response might be to simplify them, the reality is far more challenging. Our user experience professionals work tirelessly to streamline design and improve usability. Still, these products often serve multiple functions, support diverse applications, and integrate with other systems—adding layers of complexity that cannot simply be designed away.

Our professional product documentation teams play a crucial role in helping users install, navigate, and maximize the value of our products. However, documentation has its limits. Since most documentation is written during the product development phase, it covers the "happy path"—ideal use cases and known issues at launch. It cannot always anticipate real-world problems and unique edge cases that arise post-release.

The Role of Technical Support

When users encounter challenges, they turn to online help resources, contact technical support, or submit support tickets. This process is both resource-intensive and costly for businesses, requiring substantial investment in support teams. Many organizations structure their support operations in tiers:

- Level 1 Support: Initial issue screening and basic troubleshooting.
- Level 2 Support: Deeper problem analysis for more complex cases.
- Level 3 Support: Engineering-level debugging and issue resolution.

The deeper a ticket progresses through these tiers, the more expensive it becomes—for both the company and the customer. Beyond improved product design, customer self-service is key to managing these costs.

The Evolution of AI in Support

Many technical support teams have modernized their approach over the years:

- Embedding product documentation within ticketing systems instead of maintaining separate help sites.
- Deploying AI-powered decision-tree chatbots to assist users in troubleshooting issues.
- Integrating generative AI chatbots and AI agents into support portals and ticketing systems.
- Proactively surfacing relevant content as users describe their issues, helping them resolve problems before submitting a ticket.

These advancements significantly reduce ticket volume and improve customer experience. However, one particularly persistent and costly challenge remains: repetitive support tickets.

By leveraging generative AI and automation, we can revolutionize how we address recurring issues, reduce support costs, and improve customer satisfaction. The future of technical support lies in harnessing AI to anticipate, intercept, and resolve problems before they escalate into costly support cases.

Knowledge Centered Support

There's no poorer name to describe what Knowledge-Centered Support is and isn't. As discussed, the primary documentation about installing, configuring, and using products is written during product development. However, things go wrong after product release that cannot or were not anticipated during product development.

When customers encounter such problems, they contact customer support, and here's the kicker – it isn't unusual for multiple customers to experience the same problem and duplicate support tickets. It is also common for support tickets to be fielded by different support agents. The volume of these repetitive and recurring support tickets adds up quickly and comprises a significant percentage of service requests.

So, how can shops significantly reduce repetitive service requests, enable customer self-service, and deflect recurring support tickets? Enter **Knowledge Centered Support**.

Not to be confused with a knowledge center help portal, Knowledge Centered Support (KCS) is a customer support process. The key is the suffix – Knowledge-Centereded, meaning apply the knowledge gathered during the support process and then refine and reuse it to deflect recurring support requests. Rather than rely solely on search in a complex support ticketing database for previously resolved support cases, KCS aims to provide clear and concise technical articles that help customers resolve recurring issues or assist support agents with problems that customers fail to resolve themselves. Thus, the key goals of KCS are:

- Move away from reliance on searching through complex databases of resolved cases.
- Create reusable, clear, and concise knowledge articles that either help customers directly or assist support agents.
- Continuously improve knowledge quality through contributions and feedback during the support process.
- Utilize KCS articles as a significant source of material for AI-support chatbots and agents.

But there's a fly in the ointment, as they say. The support agent or engineer analyzes and resolves new support requests in many shops as quickly as possible. When we divert them to write technical support articles, they're not servicing customers, and if they don't stop and write KCS support articles, they can't service new requests. That's a problem and a common reason why implementing KCS often fails.

Ideally, the professional technical documentation team would write KCS articles for several reasons.

- They're experts at writing clear, concise, and consistent content.
- They wrote the original product help documentation work directly and daily with the most knowledgeable technical experts – the engineers that developed the products and can reuse their knowledge and the existing content while developing KCS articles – ensuring consistency
- They can update, correct, and extend the existing help content – all while freeing up support agents and engineers to service new support requests.

The Challenge: Bridging the Gap in KCS

While many support agents are not professional technical writers, they often have a unique advantage when creating Knowledge-Centered Support (KCS) articles. Having worked directly on the original issue, they become instant subject matter experts (SMEs) with firsthand knowledge of the problem. They also have direct access to case records, customer interactions, and detailed troubleshooting steps that traditional technical writers may not have in real-time.

However, significant challenges arise: Support agents are tasked with resolving cases in minutes, and leadership wants their attention focused on resolving customer issues – not diverted to writing support articles. The separation of support and content development systems, with limited cross-access, adds complexity. This siloed structure can slow the flow of valuable knowledge between support and documentation teams, making it difficult to capture and share insights efficiently.

The Need for Automation in KCS

To bridge this gap, the process must be streamlined and automated to facilitate cross-functional collaboration. Without automation, KCS can be a time-consuming and costly initiative. A well-structured enterprise KCS requires input from multiple stakeholders, including:

- **Support agents** logging and documenting issues.
- **Subject matter experts (SMEs)** providing deeper insights.
- **Developers and product teams** validating solutions.
- **Technical writers** refining and structuring content.
- **Reviewers and approvers** ensuring accuracy and compliance.

Beyond content creation, KCS governance is complex, involving:

- **Task management** – Assigning roles and tracking progress.
- **Access control and authorization** – Managing who can create, edit, and publish content.
- **Editorial and technical reviews** – Ensuring accuracy and clarity.
- **Localization** – Translating content for global users (if needed).
- **Publishing workflows** – Distributing content across multiple channels.
- **Lifecycle management** – Periodic content reviews, updates, and eventual retirement.

Without automation and structured workflows, KCS programs can become unwieldy, leading to inefficiencies and failure.

The Scale of the Challenge

For companies managing dozens or hundreds of products, the scale of KCS content creation is immense. It's common for support teams to generate many KCS article requests per week, amounting to hundreds or thousands of KCS articles. In large enterprises, the number of active KCS articles can reach tens or even hundreds of thousands.

This sheer volume highlights the urgent need for AI-driven automation in KCS processes. By integrating intelligent workflows, AI-assisted content generation, and seamless collaboration tools, organizations can turn support knowledge into a powerful, scalable asset—reducing repetitive tickets, improving self-service, and enhancing overall customer satisfaction.

Automating KCS Workflow

Most ticketing systems are powerful platforms where support agents manage cases from initiation to resolution. These systems facilitate the capture and sharing of information internally and with customers. By integrating an automated KCS workflow, generating support articles can be streamlined to a simple button click.

KCS Article Request Initiation

When a support agent resolves a case and identifies an issue likely to generate multiple future requests, they can initiate a KCS article request directly from the support ticket. Clicking the request button triggers an automated process that routes the request to a central content planning system.

Because completed support tickets already contain extensive metadata, case details, and communication history, the goal is to minimize additional input from the support agent. The workflow automatically extracts this data—referred to as the **support case payload**—to streamline the content creation process.

KnowledgeArticleRequestFlowForCase	
* Case Number	24512026
URL	https://aval
Case Owner Name	Alex
Case Owner Email	alex
Subject	Help
Email Address (watchers)	
Upload a file	<input type="button" value="Upload Files"/> Or drop files
List Of Approvers	
* Issue Category	--None--
Article Type	--None--
Functional Description	

Figure 1. Sample KCS Article Request from the Support Ticketing System

Automated Creation of a Persistent Digital Tracking Record

Once a KCS article request is initiated, a **digital tracking record** is automatically generated within a centralized content planning and management system—let’s call it **Content Central**. Each record represents a unique KCS article and contains all the necessary metadata to manage its lifecycle, whether it spans weeks, months, or years.

This structured approach ensures content is systematically tracked, updated, and maintained. While some may wonder why a ticketing system like JIRA isn’t sufficient, the distinction is crucial:

- **JIRA functions as a *transient* System of Record (SoR)**, primarily designed to track tickets and development tasks resolved over time.
- **Content Central is a *persistent* SoR**, maintaining active records throughout the entire content lifecycle, from creation to retirement and archival.

Organizations can efficiently manage content workflows, maintain consistency, and optimize knowledge distribution across teams using a dedicated content management system like Content Central.

Basic details	
Case Number 24511500	Status NEW
Case URL https://ava	Assigned Publication Manager Michael
Name Support article for ECS Essentials	Case Owner Name Zak
Approvers ash	Case Owner Email ash
Existing KC Article -	Approval Message -
Issue Category content	Published Date (Dev) -
Article Type internal	Published Url (Dev) -
Functional Description A test func description	Published Date (Prod) -

Figure 2 Sample KCS article publication management record in Content Central

AI-Powered KCS Article Generation

When an agent requests a KCS article, the workflow automatically generates a draft using the support case payload. This draft is then stored in the content management system, with a reference link logged in Content Central for tracking. Additionally, the AI-generated article is published to the development instance of the content delivery system, enabling easy access for review. A link to the

published draft (available in the development staging system but not on the public-facing site) is also recorded in Content Central, ensuring seamless traceability.

KCS Article Request Triage

When a record is created in Content Central, the workflow automatically notifies a content planner, who evaluates the request and determines the next steps. Requests may be rejected with a reason, returned to the submitter for modifications, or assigned to the appropriate reviewer responsible for refining the AI-generated article. If the designated individual is a writer responsible for related publications, they can also update existing help content as needed.

Writing Assignment Ticketing

The content planner assigns a writer within the Content Central publication record. Upon assignment, the workflow automatically creates a task in JIRA (or a similar ticketing system) and notifies the writer with a direct link to the ticket. This ticket moves through various stages throughout development—from Open to Resolved.

Note that the content planner can assign the writing/curation to various people. For example, an assignee can be a professional product developer, a learning developer of the content involved in courseware, or a marketing professional for pre-sales content defects (such as erroneous system requirements, support agents, SMEs, and so on. Content Central permits the content planner to select the content class, and the workflow automatically manages the creation of assignments and notifications.

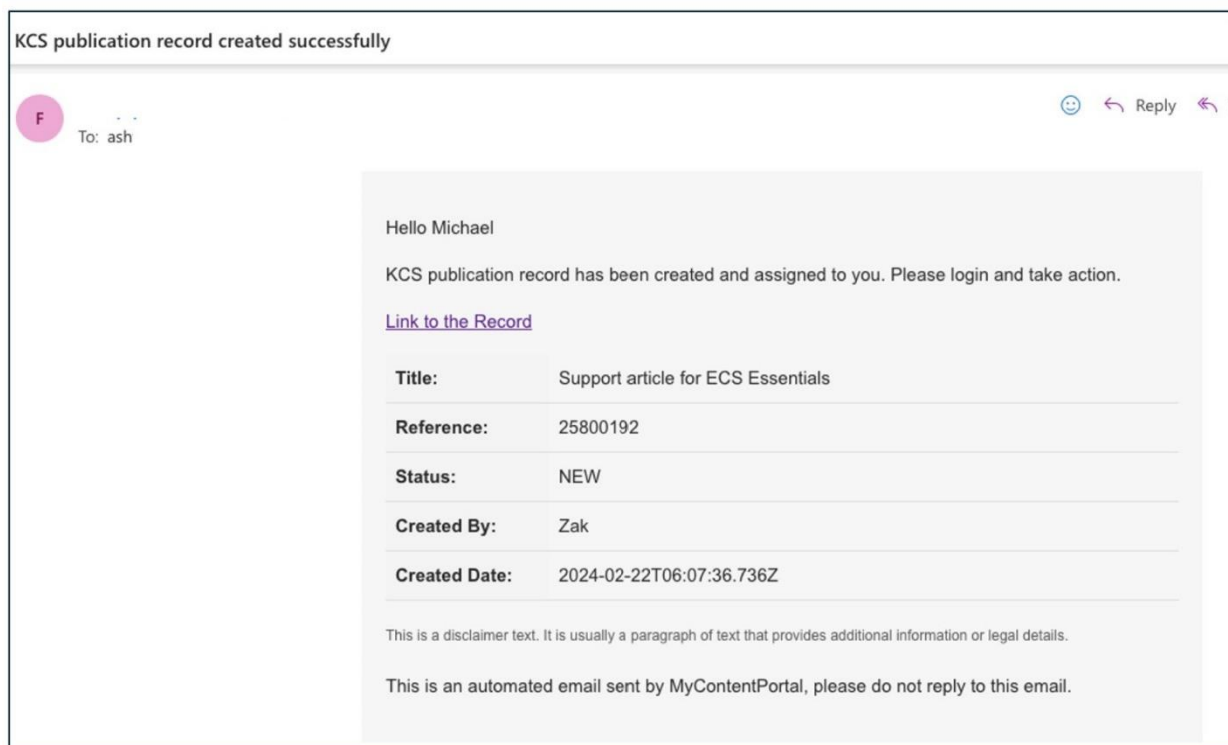


Figure 3 Sample writing assignment email notification

The JIRA ticket consolidates all necessary information for the writer, including links to the case payload, article requester, subject matter experts (SMEs), and other relevant metadata. These details are also synchronized with the publication record in Content Central. Any updates made in either system are automatically reflected in both JIRA and Content Central to ensure consistency.

Once the writer completes the article, they update the JIRA ticket status. The workflow then automatically republishes the refined KCS article to the content delivery system's development (staging) instance, logs the published article link, and notifies all key stakeholders, including the original requester, content planner, and SMEs, for final review.

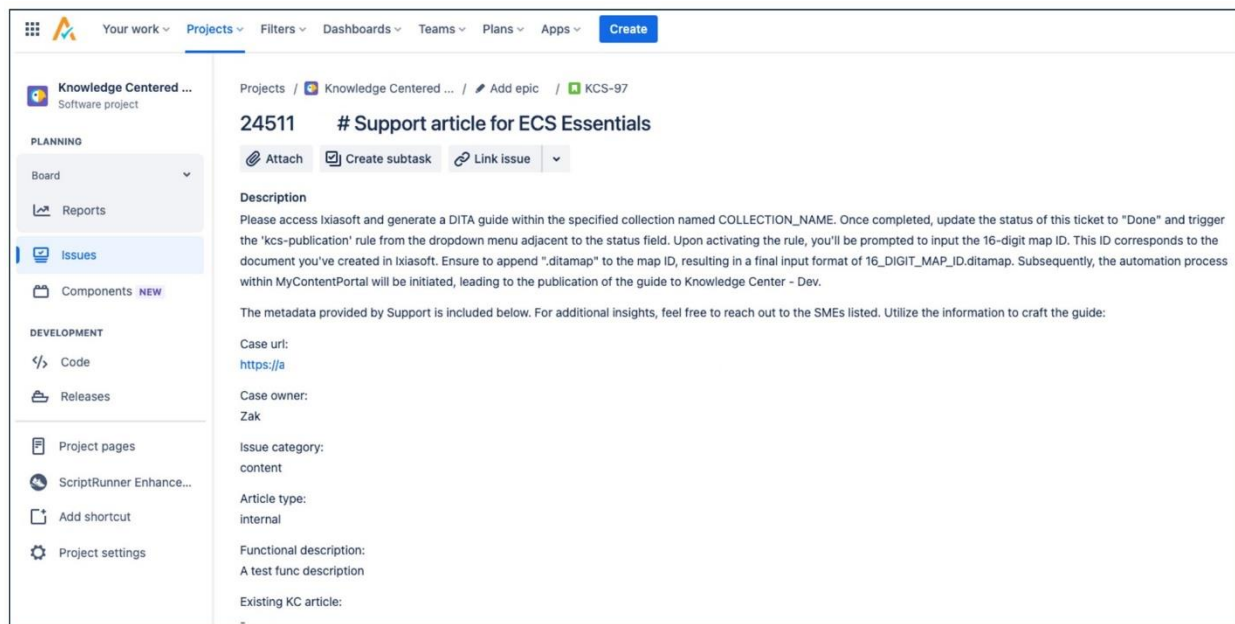


Figure 4 Sample auto-generated JIRA writer assignment ticket

Final Review, Approval, and Publishing

Final reviewers and approvers are designated within Content Central. When a writer submits a completed KCS article, the assigned approver is automatically notified via email. This email includes an embedded user interface control, allowing the approver to approve or reject the article with feedback.

If the KCS article is approved, the publication planner will be notified and will publish it in the production delivery portal(s). The workflow then updates the Content Central record and JIRA ticket with the published article link, automatically notifying all relevant stakeholders.

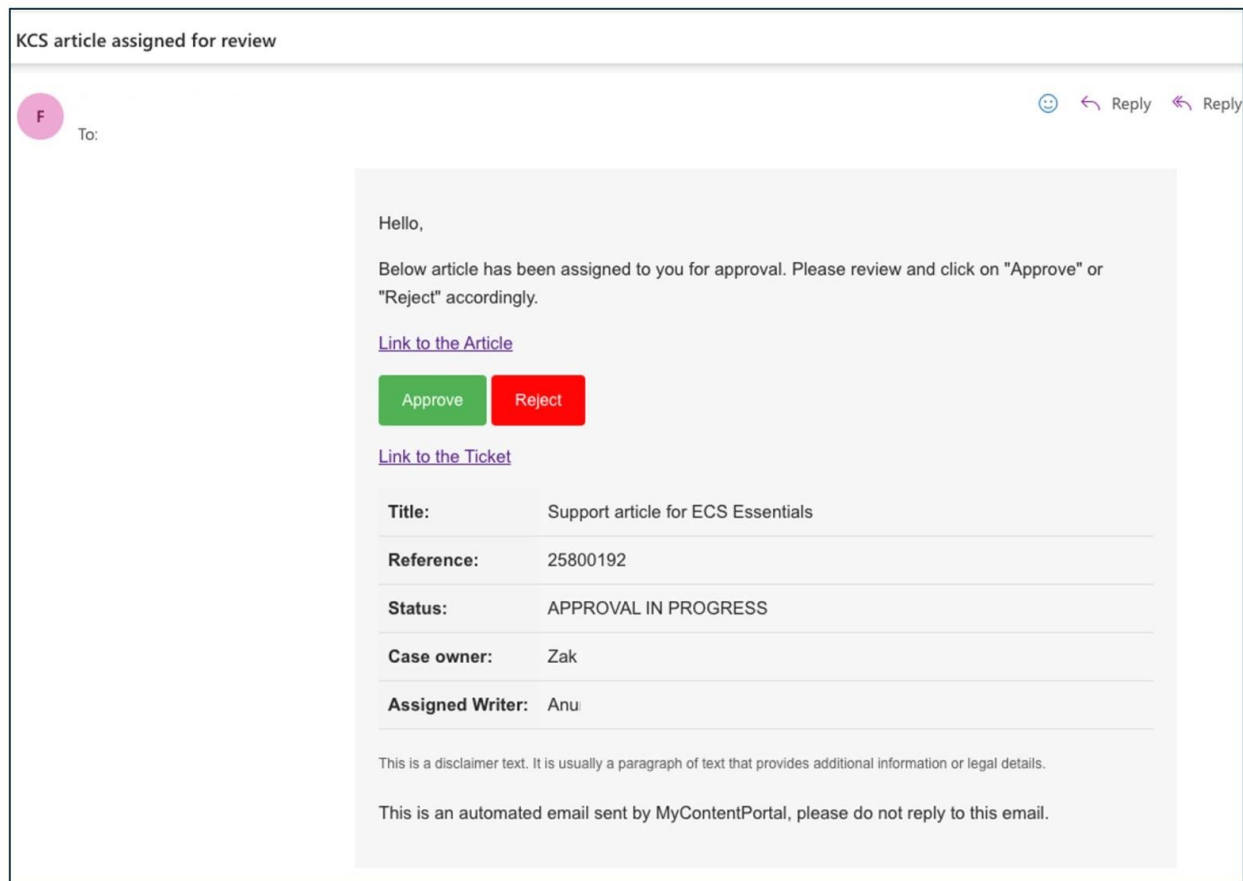


Figure 5 Sample final publishing approval email notification

Periodic Review and Retirement

KCS articles vary in lifespan. Some remain relevant for years, such as those addressing frequently recurring issues like password resets—while others serve temporary needs, such as workarounds for product defects or pending design changes.

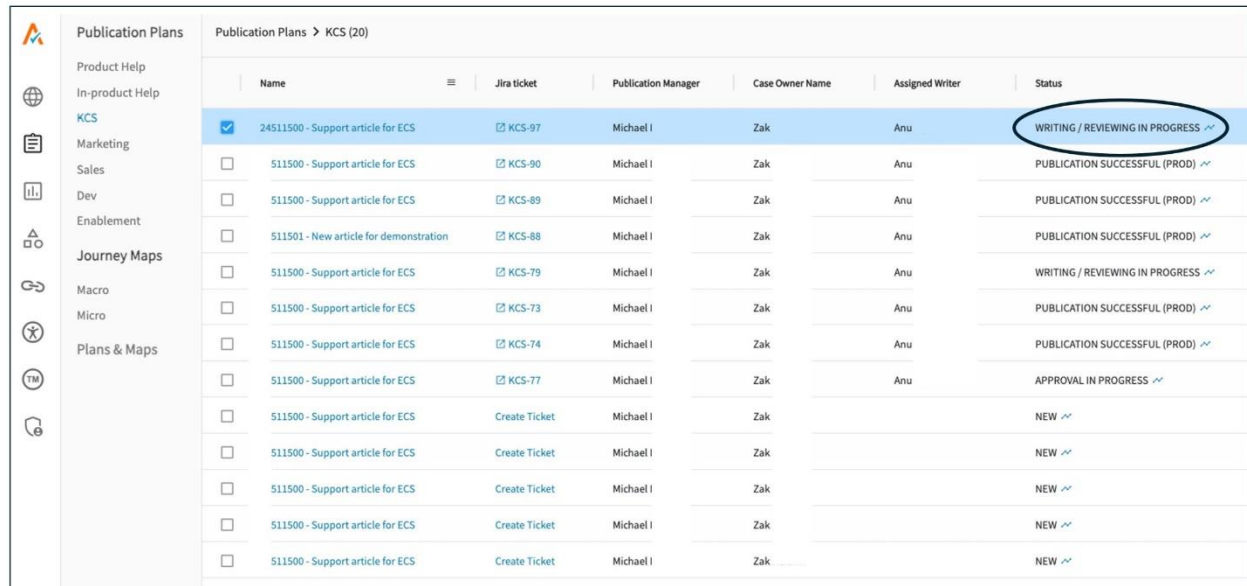
Each KCS article record includes a default review and retirement date to manage the content lifecycle, which can be modified. Articles approaching their expiration date must be reviewed and either updated or extended. They are automatically retired and archived if no action is taken, preventing content obsolescence and minimizing unnecessary content accumulation.

Content Management Dashboard and Analytics

A core feature of this system is a centralized dashboard that allows authorized participants to track the status of all article requests in real time. The dashboard provides sortable fields, direct links to individual records, and actionable controls for content planners, enabling efficient management and oversight.

Content Central also captures and presents detailed analytics for KCS articles. These insights help track article usage across content delivery platforms, log article access frequency, and measure

how often articles contribute to support request deflection. Additionally, for AI-generated KCS articles, the system can analyze the percentage of AI-generated content modified by writers, offering a benchmark for AI quality and enabling continuous refinement of automated content generation.



Publication Plans		Publication Plans > KCS (20)				
	Name	Jira ticket	Publication Manager	Case Owner Name	Assigned Writer	Status
<input checked="" type="checkbox"/>	24511500 - Support article for ECS	KCS-97	Michael I	Zak	Anu	WRITING / REVIEWING IN PROGRESS ✓
<input type="checkbox"/>	511500 - Support article for ECS	KCS-90	Michael I	Zak	Anu	PUBLICATION SUCCESSFUL (PROD) ✓
<input type="checkbox"/>	511500 - Support article for ECS	KCS-89	Michael I	Zak	Anu	PUBLICATION SUCCESSFUL (PROD) ✓
<input type="checkbox"/>	511501 - New article for demonstration	KCS-88	Michael I	Zak	Anu	PUBLICATION SUCCESSFUL (PROD) ✓
<input type="checkbox"/>	511500 - Support article for ECS	KCS-79	Michael I	Zak	Anu	WRITING / REVIEWING IN PROGRESS ✓
<input type="checkbox"/>	511500 - Support article for ECS	KCS-73	Michael I	Zak	Anu	PUBLICATION SUCCESSFUL (PROD) ✓
<input type="checkbox"/>	511500 - Support article for ECS	KCS-74	Michael I	Zak	Anu	PUBLICATION SUCCESSFUL (PROD) ✓
<input type="checkbox"/>	511500 - Support article for ECS	KCS-77	Michael I	Zak	Anu	APPROVAL IN PROGRESS ✓
<input type="checkbox"/>	511500 - Support article for ECS	Create Ticket	Michael I	Zak		NEW ✓
<input type="checkbox"/>	511500 - Support article for ECS	Create Ticket	Michael I	Zak		NEW ✓
<input type="checkbox"/>	511500 - Support article for ECS	Create Ticket	Michael I	Zak		NEW ✓
<input type="checkbox"/>	511500 - Support article for ECS	Create Ticket	Michael I	Zak		NEW ✓
<input type="checkbox"/>	511500 - Support article for ECS	Create Ticket	Michael I	Zak		NEW ✓

Figure 6 Sample content management dashboard for all KCS articles

Continuous Improvement

This overview does not cover every aspect of exception handling, such as reassignment workflows or updates to existing articles, but the framework provides a solid foundation.

Further opportunities exist for enhancement and automation. For example, organizations leveraging AI-powered content governance tools like Acrolinx can integrate them into the workflow. These tools analyze content for consistency with style guides, enforce preferred terminology, improve clarity, check grammar and spelling, optimize searchability, and enhance editorial quality. The AI article generator could automatically process its drafts through such an analyzer, applying suggested corrections before human review, streamlining the content development process even further.

Summary

Enterprise-grade software products are inherently complex, requiring extensive documentation and support resources to help users navigate challenges. While technical documentation provides valuable guidance, it often focuses on ideal scenarios and cannot fully anticipate real-world issues. As a result, businesses rely heavily on tiered technical support structures, which can be costly and inefficient, mainly when dealing with repetitive and recurring service requests.

Knowledge-Centered Support (KCS) offers a structured approach to capturing and reusing support knowledge, helping reduce ticket volume and improve customer self-service. However, traditional

KCS implementations often struggle due to the high demands placed on support agents, lack of cross-platform integration, and inefficiencies in content development.

Generative AI and automation can transform KCS by streamlining article creation, integrating support and content systems, and improving workflow efficiency. By leveraging AI-powered automation, organizations can:

- Enable support agents to initiate KCS article requests with minimal effort.
- Automate draft generation using AI, reducing the burden on support teams.
- Establish centralized content management for tracking, review, and governance.
- Facilitate seamless collaboration between support agents, technical writers, and SMEs.
- Implement lifecycle management to maintain content accuracy and relevance.

A well-executed AI-driven KCS framework reduces the reliance on expensive human resources, enhances self-service options, and improves overall customer satisfaction. Organizations can turn support knowledge into a scalable asset by integrating intelligent workflows, automated content governance, and real-time analytics, significantly improving operational efficiency and user experience.

Appendix

Workflow Summary

1. **KCS Article Request Initiation**
 - Support agent resolves a case and identifies a recurring issue.
 - Clicks a button to request a KCS article.
 - Metadata and case details are automatically extracted.
2. **Automated Content Tracking**
 - A digital record is created in *Content Central* for lifecycle management.
 - Metadata and case history are stored for reference.
3. **AI-Powered Draft Generation**
 - AI generates a draft using support case details.
 - The draft is stored and linked to *Content Central*.
 - A preview version is published in the staging environment.
4. **Triage & Assignment**
 - Content planners review requests and assign tasks.
 - Writers, SMEs, or other stakeholders refine the AI-generated draft.
 - JIRA tickets or equivalent tasks are created and tracked.
5. **Review & Approval**
 - Stakeholders review and approve/refine the article.
 - AI-assisted editing tools ensure clarity, consistency, and compliance.
6. **Publishing & Distribution**
 - Finalized article is published to support portals.
 - Links and records are updated in *Content Central* and JIRA.
 - Stakeholders are notified.
7. **Lifecycle Management & Analytics**
 - Articles have scheduled reviews and expiration dates.
 - Automated tracking ensures updates or retirements as needed.
 - Analytics track usage, impact, and support ticket deflection rates.

Disclaimer: The views and opinions expressed in this document are solely those of the author and do not represent the views, positions, or policies of Avalara Inc.