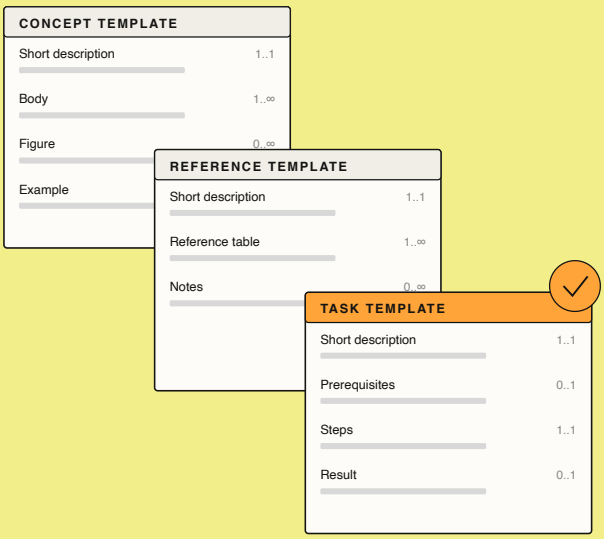
STRUCTURED AUTHORING

Set up Author-it like DITA

Concept, task and reference topics, enforced by rules, without XML.



Author-it
For Author-it Cloud 2026.R1 and later
November 2026

DITA's information model is a good one. You don't need an XML toolchain to use it. This guide shows how to build structured templates for each information type, write the rules that enforce them, gate approval and publishing on validation, map the rest of DITA onto Author-it, and convert an existing library without disrupting translation.

Contents

1 Who this is for
2 DITA to Author-it: the map
3 Plan your information types
4 Styles and structured templates
5 Rules for concept, task and reference
6 Example topics and enforcement
7 Maps, reuse and conditions
8 Convert existing content
9 Roll it out
10 Checklist

01

Who this is for

Three kinds of team ask us how to make Author-it behave like DITA. This guide is for all three.

LEAVING DITA

You're moving from a DITA system and want to keep the information model your writers know, without the XML editor and toolkit.

TOLD YOU NEED DITA

Someone said DITA is what makes content AI-ready. You want the rigour, the typed topics and the validation, without the overhead.

TIGHTENING UP

Your Author-it library grew up on ordinary templates. Topics of the same type no longer look the same, and you want that fixed for good.

What you will have by the end

- A structured template for each information type, starting with concept, task and reference.
- Rules that say what each type must contain, in what order, and which styles and objects it may use.
- A release state that only accepts content that validates, and publishing profiles that only take that state.
- A clear map from DITA features to the Author-it features that do the same job.
- A plan for converting existing content without triggering re-translation of everything you have approved.

What this guide is not

It isn't a DITA migration service, and it doesn't produce DITA XML. Author-it content is stored as objects in a relational library and exports to XML, but it isn't DITA. If a customer, regulator or partner requires DITA interchange, talk to us about that separately.

Everything here uses features documented in the Author-it Cloud 2026.R1 Author Module Guide and on the Author-it Knowledge Center. Style, template and release state names are examples. Use your own.

WORTH KNOWING

Structure rules in Author-it are opt-in, per template. A library where nobody switched them on can look free-form. That's a configuration choice, and this guide is how you change it.

SUMMARY

DITA's discipline, an editor your writers already know.

02

DITA to Author-it: the map

Most of what DITA gives you has a direct equivalent in Author-it. Here is the map we use with customers.

IN DITA	IN AUTHOR-IT
Topic types: concept, task, reference	A structured topic template per type, with rules in the Structure Builder
DTD or schema validation	Rule validation in the Editor (Rules tab, Validate Topic, Validate Book), plus a release state that enforces it
shortdesc	A required, fixed first paragraph rule using a Short description style
steps, step, cmd, stepresult	A required group rule built from Step and Step result paragraph rules
Map and submap	Book and sub-book, with book structure rules
Front and back matter in a bookmap	Title page, table of contents, table of figures, glossary and index rules in a book structure
conref and conkeyref	Embedded topics, restricted by an embedded topic inline rule
keyref for text and values	Variables (text, list of values, file), resolved at topic, book or library level
Conditional attributes and DITAVAL	Variants, variant criteria and filtering when you publish
xref and related links	Hyperlink objects, which resolve through variants and publishing
Specialisation	A copy of a structured template with changed rules
DITA-OT transforms	Publishing profiles: HTML, Magellan portal, Word and PDF, XML, AION JSON for AI
Workflow (not part of DITA)	Release states, permissions and the Reviews module

Two honest differences

Author-it content isn't portable DITA XML. It exports to XML, and translation jobs travel as XML with a schema, but if you need to hand DITA files to a partner, DITA is the better tool for that job.

Rules are set per template rather than inherited through a specialisation chain. That's simpler to run day to day. It means each type's rules live in its own template, so changes to a shared pattern are made in each template that uses it.

03

Plan your information types

Start with an audit, not the Structure Builder. Rules written before you know your content get rewritten.

1 Group what you have

Take a representative set of books and sort their topics into types. Concept, task and reference cover most technical documentation. Troubleshooting, glossary and safety content are common additions. Don't invent types you have no content for.

2 Write each type down in plain words

For each type, list what it must contain, what it may contain, the order, and anything it must never contain. "A task starts with one short description, may have prerequisites, must have numbered steps, and may end with a result. No images in the short description." That sentence becomes your rules.

3 List the styles each type really uses

Check which paragraph and character styles appear in each type today. Standardise them now. Rules validate against styles, so a type that uses six different heading styles will fail until you pick one.

4 Decide where validation is enforced

Pick the release state that will require valid content, usually your approved state. Content can be drafted and reviewed freely before that point.

5 Decide how to convert

Either convert existing templates, which applies the structure to every topic based on them at once, or create new structured templates and move topics across as they come up for review. Most teams with large libraries choose the second.

IF YOU TRANSLATE

Create a new release state for enforcing validation rather than reusing the one that sends content to translation. Converted topics then don't get released for re-translation.

SUMMARY

Types, rules in plain words, styles, one release state, a conversion route.

04

Styles and structured templates

Author-it already requires styles instead of direct formatting. Structure rules build on that, so get the styles right first.

Create the styles each type needs

Typical additions for a DITA-style model are Short description, Prerequisites heading, Step, Step result, Result, Example heading and a reference table style. Styles are objects in the library, managed centrally like everything else.

Put standards where only standards owners can change them

Keep styles and templates in their own folder, and use Folder Action Permissions (File > Folder Action Permissions, maintained by your Security Administrator) so only the people who own the content model can create or change them. Everyone else uses them.

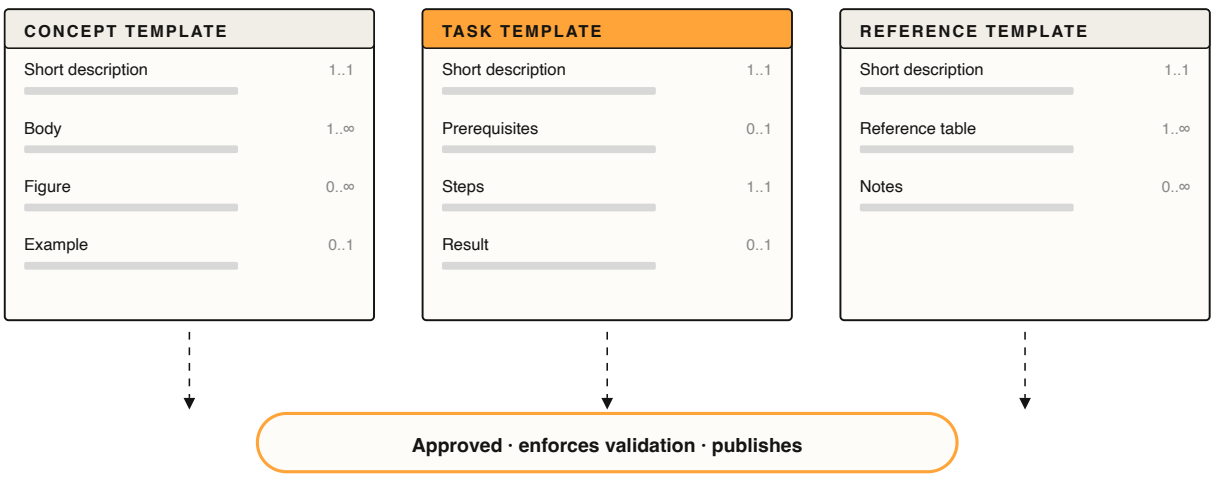
Make each template structured

- Create one topic template per information type, for example Concept, Task and Reference.
- In the template's properties, select the check box that makes it a structured template.
- A template must be based on "(none)" when you make it structured. Once it is structured, you can base it on another template.
- Changing an existing template applies its structure to every topic based on it. If those topics are already in your enforcing release state, Author-it asks you to move them, save the change as a copy of the template, or cancel.

Book templates too

Create book templates for each kind of deliverable, such as User guide, Installation guide or Release notes. You'll give them structure rules in section 7. Book templates can also be restricted to the publishing profiles they're built for, so nobody publishes a print book to a portal profile by mistake.

Fig 2.0



One structured template per information type. Rules set what is required, in what order.

05

Rules for concept, task and reference

Select the template, open **Manage > Data > Structure Builder**, name the structure on its root, and add rules.

The rule kinds you'll use

- **Paragraph** rules say which paragraph styles may appear at that point in the topic.
- **Group** rules wrap several rules into a section, such as Steps, that repeats or is optional as a unit.
- **Table** rules, with table cell rules, control the styles used in rows and columns.
- **Inline** rules sit under a paragraph rule and restrict character styles, embedded topics, images, hyperlinks and variables, or enforce instances of text.

The properties that make a rule strict or loose

PROPERTY	WHAT IT DOES
Minimum and maximum	1 and 1: exactly once. 0 and 1: optional. 1 and Unlimited: at least once. 0 and 0: prohibited, for example no images in a short description.
Sequence	Fixed rules must appear in the order they sit in the structure. Flexible rules can swap with neighbouring flexible rules in the same group.
Allowed objects	Specific styles or objects, or a template, so anything based on it is valid. In a structured topic, authors only see the paragraph styles the rules allow.

Concept

RULE	ALLOWED	OCCURS	SEQUENCE
Short description	Short description style; inline image rule set to 0 and 0	1 to 1	Fixed
Body	Body Text, List Bullet, List Number, Note	1 to Unlimited	Fixed
Figure	Figure, Caption	0 to Unlimited	Flexible
Example (group)	Example heading, then Body Text or Code	0 to 1	Flexible

Steps styles are simply not in the allowed list, so a concept can't contain a procedure. That is the rule most libraries are missing.

Task

RULE	ALLOWED	OCCURS	SEQUENCE
Short description	Short description style	1 to 1	Fixed
Prerequisites (group)	Prerequisites heading, then List Bullet	0 to 1	Fixed
Context	Body Text	0 to 3	Fixed
Steps (group)	Step (1 to Unlimited), Step result, Note; embedded topics limited to the Warning template	1 to 1	Fixed
Result	Result style	0 to 1	Fixed

Reference

RULE	ALLOWED	OCCURS	SEQUENCE
Short description	Short description style	1 to 1	Fixed
Reference table	Table rule with heading row and Table Text cells	1 to Unlimited	Flexible
Notes	Body Text, Note	0 to Unlimited	Flexible

Design tips from the product guide

Give every optional group its own opening style

If several optional groups start with the same heading style and use the same body styles, validation can't tell which group an author meant. It matches the first rule that fits, then reports a required group as missing. A Prerequisites heading style and a separate Example heading style avoid that.

Or merge look-alike groups

Where sections genuinely share styles, combine them into one optional, flexible rule rather than several near-identical ones.

Duplicate to specialise

To create a variant of a type, such as a Maintenance task, select the structured template and use Manage > Versioning > Duplicate, then change the copy's rules. The copy keeps the template settings and structure.

REMEMBER

These tables are a starting point. Your style names, occurrences and groups should come from the plain-words description you wrote in section 3.

SUMMARY

Required, optional, prohibited. Fixed or flexible. Allowed styles.

06

Example topics and enforcement

Rules tell authors what's wrong. Examples show them what's right. Release states make it stick.

Link an example topic to each structure

Write a finished example of each type, with notes for authors if you like, and save it anywhere in the library. Best practice is not to base it on the structured template itself. In the Structure Builder, select the structure's root, choose the Example Object button and pick the topic. Authors see it in the Editor's **Example** tab while they write.

What authors see

- The **Rules** tab shows each rule and whether the content validates against it.
- **Validate Topic** and **Validate Book** on the Review tab check on demand.
- In Quick-Editing Mode, the folder list marks structured objects as valid (tick), invalid (cross) or unvalidated (not structured). Rules themselves aren't shown there; validation is checked when a release state is applied.

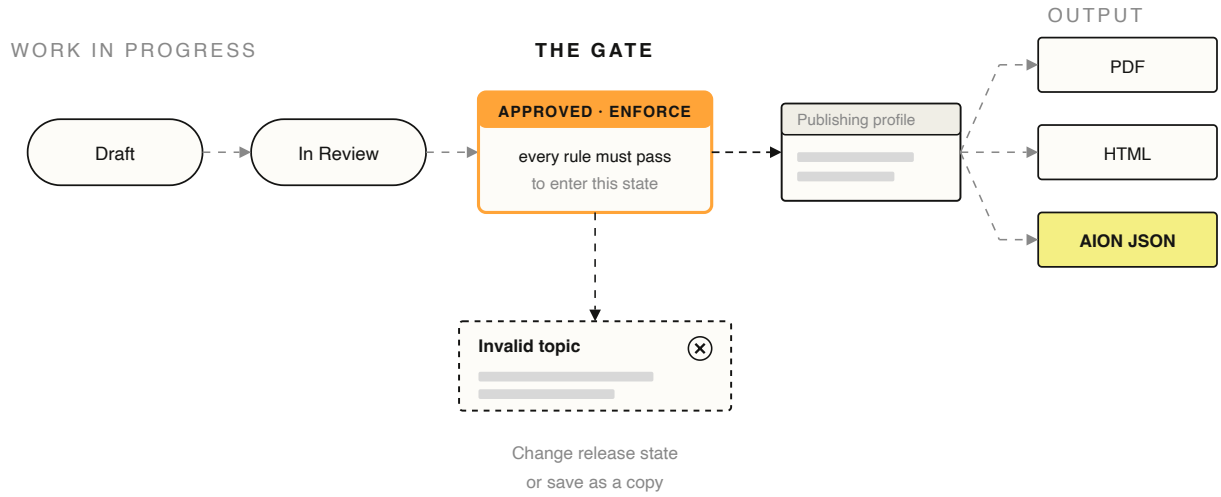
Set the enforcing release state

On the structure's root, choose the Release States button and select the state that enforces validation. Content can't be moved into that state until every rule passes. Once there, edits that would break validation can't be saved over it: the author changes the release state, saves the change to a copy (prefixed "Copy of"), or cancels.

Close the loop in publishing

Set your publishing profiles to only include content in the approved release state. With both in place, a topic must validate to publish. That covers PDF and Word, HTML and the Magellan portal, and AION JSON for AI.

Fig 2.0



A release state that enforces the structure only accepts content that validates.

07

Maps, reuse and conditions

A DITA map becomes a book. Book templates get structure rules of their own.

A User guide book structure

RULE	ALLOWED	OCCURS	SEQUENCE
Title page	Your title page template	1 to 1	Fixed
TOC	Table of contents	1 to 1	Fixed
Overview	Topic rule: Concept template	1 to 1	Fixed
Content	Topic rules: Concept, Task and Reference templates; Book rule for sub-books	1 to Unlimited	Flexible
Index	Index	0 to 1	Fixed

Front and back matter (title page, TOC, table of figures, glossary, index) can each appear once. Allowing a template, rather than a specific object, lets authors add any topic based on it. Book Content Duplication lets the template designer say which topics may or must be reused or copied when a new book is created from the template.

Reuse: conref and keyref

- **Embedded topics** replace conref. Embed a whole topic or a fragment, and change it once. Add an embedded topic inline rule to limit what may be embedded, for example only topics based on your Warning template inside steps. Pasted text is validated like any other content.
- **Variables** replace keyref for values such as product names, model numbers and version strings. They resolve at topic level first, then book, then the library default.

Conditions: DITAVAL

Variants are parallel versions of an object, selected by **variant criteria** such as Product, Country, Platform or Audience. Filtering at publish swaps in the matching variant, so one book publishes as the X2001 guide for Australia or the X3000 guide for Germany. Variants are maintained alongside their primary object.

Cross-references

Hyperlink objects manage cross-references as objects in their own right, and resolve correctly through variants and publishing.

IN PRACTICE

Most DITA reuse habits carry straight over. The difference is that authors pick an object from the library instead of typing an ID into an attribute.

SUMMARY

Map is book. conref is embed. keyref is variable. DITAVAL is variants.

08

Convert existing content

Pilot first. Applying a structure to content that never followed one will produce failures, and that's useful information.

1 Pick a pilot

Choose a small set of books that represent your content. Apply the new templates to their topics, or rebuild them on the new templates.

2 Import into the right types

When you import Word or other files, the Import Wizard opens the Topic Structure Wizard. It shows each paragraph style found, the outline level it will create, and the topic template new topics will be based on. Map heading styles to your Concept, Task and Reference templates here, and save the mapping for next time.

3 Validate and read the failures

Run Validate Book and work through the Rules tab. Decide for each failure whether the content or the rule should change.

WHY IT FAILED	FIX THE CONTENT	OR FIX THE RULE
Style or object not allowed	Use an allowed style or object	Add it to the rule's allowed objects
Too few or too many	Add or remove content	Change minimum or maximum
Wrong place	Move it into the right order	Make neighbouring rules flexible
Outside every rule	Rewrite or remove the extra content	Add a rule, or widen an existing one

4 Use help where it saves time

From 2026.R4, Author-it Cloud customers can connect Claude to their library through Author-it MCP and ask it to check topics against your style guide and report what doesn't conform, before you validate.

EXPECT THIS

A first pass usually fails on styles and on content outside the rules. Both are the inconsistencies you set out to remove.

SUMMARY

Pilot, map styles at import, validate, fix content or rules.

09

Roll it out

Convert one type at a time, starting where inconsistency costs you most.

WEEKS 1 TO 2

Model

Audit, write the types in plain words, standardise styles, lock the standards folder.

WEEKS 3 TO 4

Pilot

Build the structured templates and example topics. Validate the pilot books. Adjust rules.

THEN

Convert

Switch on the enforcing release state. Convert templates type by type, tasks and safety content first.

Train authors on two things

The Rules tab, which tells them what's wrong, and the Example tab, which shows them what's right. Everything else about their day stays the same: same editor, same styles, no XML.

Choose where to be strict

DITA enforces typing everywhere. Author-it lets you choose. Lock down tasks, warnings and regulated content with fixed, required rules. Keep release notes looser. Leave content that doesn't need a model on ordinary templates.

When you change the rules later

Saving a changed structure validates every topic based on that template. If approved topics would become invalid, you choose: move them to another release state, save the new rules as a copy of the template, or cancel. Nothing breaks silently.

Why it matters for AI

When every topic of a type has the same required parts in the same order, retrieval and AI answers get more consistent. AION publishes approved content as structured JSON through the same publishing profiles, so the gate that keeps invalid topics out of PDF and HTML keeps them out of AI output too.

GO DEEPER

The Structured Authoring and Content Reuse Handbook covers reuse, variants and governance across the whole lifecycle. author-it.com/resources

SUMMARY

One type at a time. Strict where it counts.

10

Checklist

Plan

- [ ] Topics grouped into types
- [ ] Each type written in plain words
- [ ] Styles standardised per type
- [ ] Enforcing release state chosen
- [ ] New release state if you translate

Build

- [ ] Standards folder with restricted permissions
- [ ] One structured template per type
- [ ] Rules: occurrences, sequence, allowed objects
- [ ] Unique opening style per optional group
- [ ] Example topic linked to each structure

Enforce

- [ ] Release state set on each structure's root
- [ ] Publishing profiles take approved content only
- [ ] Book templates with structure rules
- [ ] Book templates restricted to their profiles

Convert and run

- [ ] Pilot books validated
- [ ] Import mappings saved in the profile
- [ ] Authors shown the Rules and Example tabs
- [ ] One type converted at a time

Author-it

Where to go next

author-it.com/contact

Evaluating Author-it?

Bring one topic type you care about. We'll build its structure rules on your own content in a working session.

author-it.com/contact

Already a customer?

Talk to your Account Manager about a structure workshop for your library, or read the Knowledge Center's Structured Authoring section.

kc.author-it.com

Comparing with DITA?

Our side-by-side comparison covers authoring, reuse, governance, translation, AI readiness and cost.

author-it.com/comparison-author-it-vs-dita