

# vjoon K4 6.0 Comes Close to Publishing in the Real World with Task-based Workflow



# Executive Summary

Most currently available publishing systems apply a status-based workflow in order to push layout and content objects through the publishing process. Most disturbing about status-based workflow is that it is counter-intuitive, resulting in steep learning curves, high training costs, and error-prone workflows.

A task-based workflow in which the system tells the users working with the system, what to do, instead of simply telling what the result should be as in status-based workflows is less prone to errors and has a less steep learning curve when done well.

vjoon K4 6.0 uses a task-based approach with rewinds, dependencies and the ability to have split workflows that can synchronise. K4 6.0 enables users to finish tasks without the need to check in or check out objects. The user can simply accept a task and from that point onwards the system creates the components needed to complete the task and move the user through the workflow until the desired end-result is achieved.

Unique about K4 6.0 is that it uses a simple to use GUI-based Workflow Editor effectively enabling administrators to rapidly set up extremely complex parallel cross-media publishing workflows.

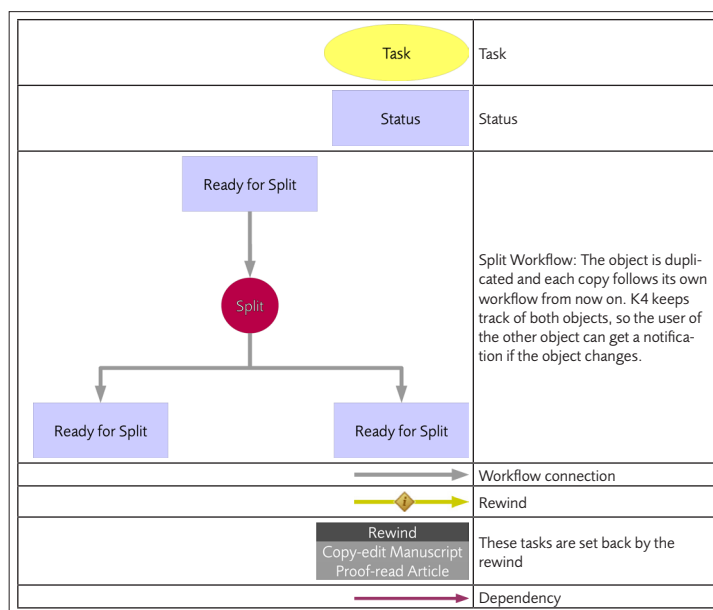


Figure 1: Legend for the illustrations on the next pages.



## Status-based Workflow

Most currently available publishing systems apply what is known as a status-based workflow in order to push layout and content objects through the publishing process. Status-based workflow depends on users selecting the appropriate status for every object they work on, with every change they make to the object. It also depends on the system offering customisable status preferences, as not all publishing environments are identical.

Status-based publishing systems use an object tracking system that requires users to change the state of an object so that the next user in line knows the object has changed and is ready.

As objects cannot be used simultaneously by multiple users, there needs to be a mechanism that locks the object as long as a user is working on it. The way this works with current systems is to require from the user that he checks out an object when he's going to work on it, and check in that object when finished.

While some systems will enable automatic status-change from “Editing” to “Ready” or “Correcting”, based on the user who has checked-in the object and the stage at which the object has been checked-in, the check-out / check-in procedure itself has to be explicitly initiated by the user in all currently available systems.

Most disturbing is that status-based workflow is counter-intuitive, and does not do a good job of mimicking the real world. In fact, Adobe—who uses the Check-in / Check-out paradigm in its Dreamweaver product—dedicates no less than a full web page explaining how this works and what happens when users check out or check in a file. ([http://kb.adobe.com/selfservice/viewContent.do?externalId=tn\\_15447](http://kb.adobe.com/selfservice/viewContent.do?externalId=tn_15447)).

Status-based systems are built around the assumption that every user in the workflow knows what to do when an object is assigned to them. When an object does not yet exist, assigning it to a user implicitly instructs the user to create the object. However, this clear-



cut and implicit task message is often less obvious when the object comes back to a user in a correction round. In those cases, the production manager needs to explain what he expects from the user (often in a note). At times when the production manager is not available to take a decision, the object's workflow is suspended until it is clear to the user what he is supposed to do. This is an important problem when freelances are involved, or the publisher depends on remote offices, perhaps working across different time zones, sharing the same decision makers.

Another flaw of the status-based workflow is that it is unclear whether objects that will be used to populate multiple output channels are ready for each of the channels they will be pushed into. For example, an image may be colour-proofed and ready for print, but it may be possible that a low-resolution, compressed JPEG for the web is not available. This is not immediately apparent from the status the object is in.

## **vjoon K4 6.0 task-driven workflow**

With version 6.0 of its publishing platform K4 for publishing environments that scale from small to large, vjoon proposes a dramatically different approach. vjoon developed a workflow paradigm that is less technical and closer to what happens in publishing environments in the real world. K4 6.0 is a publishing system that is built around task-based workflow.

In a task-based system, the system tells the editor, layout designer, journalist, writer, photographer, illustrator, or anyone else working with the system, what to do. It does not simply tell what the result should be as in status-based workflows.

To understand the difference between K4 6.0's approach and that of traditional publishing systems, consider a taxi drive. When a businessman takes a cab at the airport to drive him home, he doesn't tell the cab driver: "Take me to status 'At Home'". Instead he instructs the cabbie to drive him to Cromwell Avenue 12. The driver will take the businessman to that



address and on arrival, it is clear to both him and the cab driver the status 'At Home' has been reached (unless the cab driver didn't know the address).

In a publishing process, an editor tells the writer to write an article of 600 words on for example the credit crunch for publication on date N in magazine A. The writer writes the article and sends it to the editor. The editor checks whether the article is up to standards and sends the article back to the writer if it isn't —perhaps with instructions on how to improve— or to the next stage in the publishing stream if it meets the pre-defined criteria, which can be the layout or the output channel. This task-based stream of processes is what vjoon K4 6.0 mimics.

By applying this workflow paradigm to a digital system, the system does no longer have to show which publishing component is assigned to each user. It has to show the user which tasks he has to perform. However, at the origin of this task-based workflow lies the organisation of tasks and roles. Translated to a digital system this implies an administrator needs to set up the tasks and the roles that fit the tasks. This setup process has to be done only once and can be compared to setting up a general-purpose role-based workflow system where business policies deter-

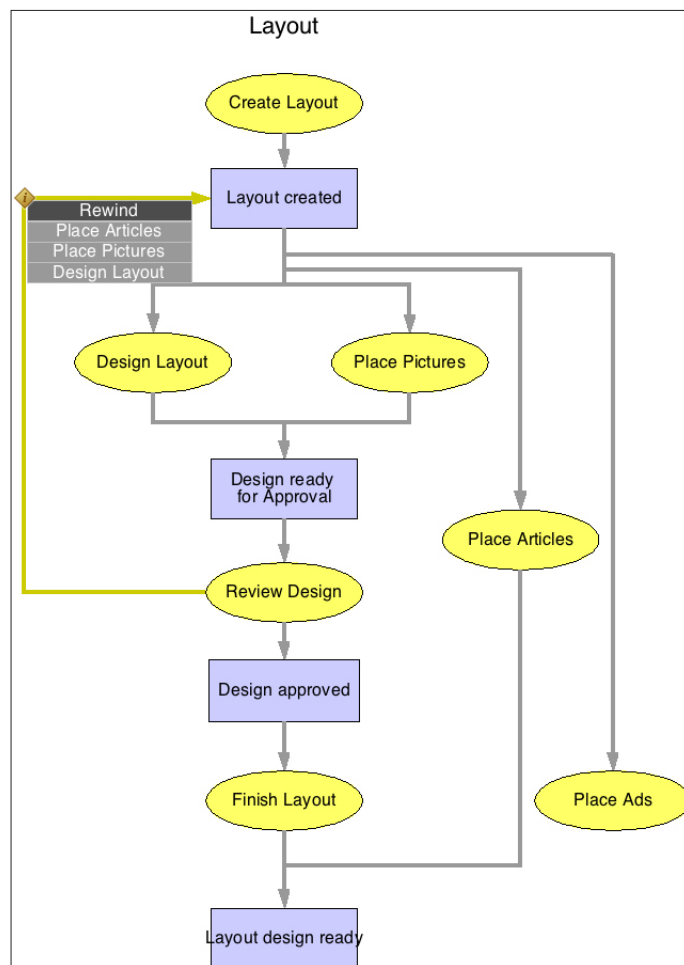


Fig. 2: Example of a rewind action in the Workflow Editor.



mine the respective roles (editors, photographers, writers, freelances) that can (and must) perform specific tasks.

## GUI Workflow Editor

To this effect, the administrator or editor who sets up the system will use K4 6.0's user-friendly Web-browser based GUI (Graphical User Interface) Workflow Editor in the K4 Admin interface.

This interface not only allows admins to set up the straight-through workflow (for the ideal situation where no problems occur on the way does not exist) but also to define 'rewinds' to help the user decide what to do next if a problem occurs.

Such a rewind may be taken quite literally, i.e. the system will rewind the workflow to the first known state before the problem occurred.

## Dependencies

One of the problems with status-based workflows is that some objects may be ready for print but not for web (or vice versa) without this discrepancy being visible. In K4 6.0, de-

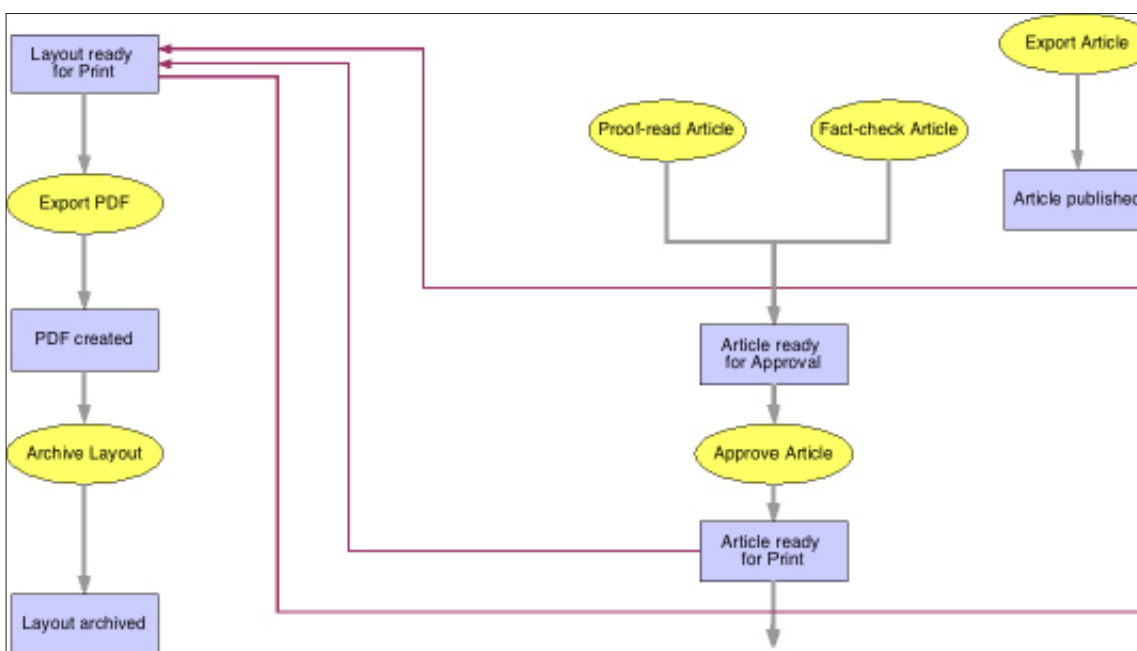


Fig. 3: Dependencies; in the example above, the "Layout Ready for Print" status is dependent on the "Article ready for Print" status as depicted by the arrows in the Workflow Editor.



dependencies help avoid errors, because a layout cannot be output as long as there are placed objects that do not have or surpass a specific status. For example, all images need to be in the “Ready for web” status before an article can be pushed into the online publishing channel.

With K4 6.0’s task-based approach, there is no longer a need for checking in or checking out objects. The user simply accepts a task and from that point onwards the system creates the components needed to complete the task.

This can be a file, a layout or an article, or only one metadata field. This also allows tasks to be completed in parallel even when they have an impact on different parts of the components.

Instead of checking in an object, the user only tells the system that he wants to fulfill a task. When finished, he has to tell the system whether he handled the task successfully or not. The system decides —based on the workflow definition— which status the object should be switched to or which alternative or complementary task has to be performed to solve the user’s problem. The exact mechanism must be defined by the administrator in the GUI Workflow Editor.

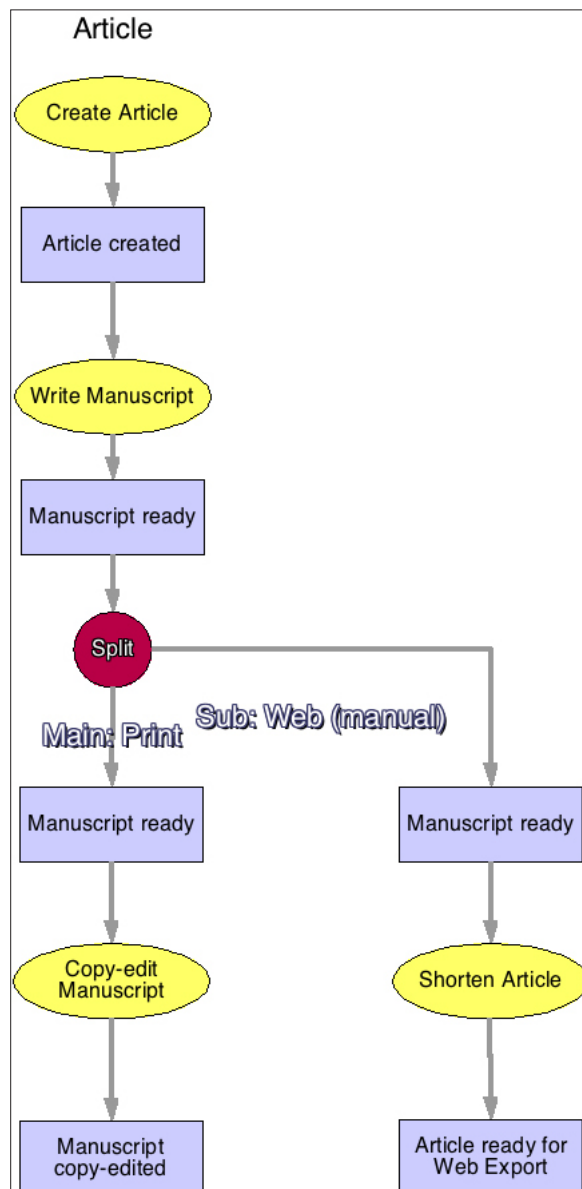


Fig. 4: A split workflow where print and web run in parallel.



The benefit is twofold. Firstly, the system ensures the responsibility for the task stays with the user performing the task from start to finish. Secondly, the systems ensures the workflow is not obstructed in any way, saving time and therefore money.

## Split Workflow Supporting Multi Channel Publishing

The Split Workflow is another major enhancement that allows users to send an object to multiple output channels including print, mobile, and online. Split Workflow enables true cross-media publishing. K4 6.0 will split the object in as many versions as needed, tracking all versions and keeping them in synch, unless you explicitly break the synchronisation.

The aim is not only that more than one article could result from the split, but also to use the same content for multiple editorial departments from an early state in the workflow. In addition to the synchronisation functionality, therefore, if one of the documents that resulted from a split has been changed, the user of the alternative document can receive a notification.

In a cross-media publishing workflow, the author may write to a specific limit for one channel and another limit for another channel. If, for example, the web site has been designed so that each article may contain a maximum of 400 words for online best readability, and the printed edition may contain 1200 words because that's how many words fit on two pages, the author will probably first write the lengthy version first, and shorten this version afterwards to accommodate the shorter version for web usage. Split Workflow and Dependencies ensure that all output media get the finished objects automatically before these are output to their respective channel.

## Conclusions

Task-based workflows are much more intuitive to use, and are therefore also more comfortable to work with by freelancers or —where Corporate Publishing is concerned— account-



ants who input financial data in an annual report. Thus, the major benefit of a task-based approach is that it requires less training and has a less steep learning curve altogether.

Users no longer need to know which publishing component is assigned to them. The K4 6.0 system can simply show them which tasks they have to perform. The complexities of the publishing workflow are moved away from the user to the administrator level. Administrators can deal much better with these complexities as managing workflows belongs to their core knowledge.

Even then, the K4 6.0 Workflow Editor being GUI-based is simple to use and enables rapid set-up of complex multi-channel workflows, effectively supporting cross-media publishing needs.

The system makes it clear-cut who must do what, and ensures tasks can be performed without running into obstacles. And when problems occur with the content or design itself, rewinds enable the workflow to 'step back' allowing the user to start over again or correct the errors he has made.

The K4 6.0 workflow engine allows publishers to define as many workflows as they need, such as different workflows for photos, infographics, etc.

Despite having different workflow definitions, K4 6.0 still features dependencies between them, which allows for a more integrated approach of the publishing process than what current publishing systems allow for.

