Workflow performance improvement and load reduction

1 Contents

2 Conceptual overview

Nintex Workflow utilizes the SharePoint workflow engine, which in turn is built on the Windows Workflow Foundation. Like with many aspects of SharePoint, there are certain performance limitations and common errors that can be encountered, not all of which are documented. This document has been compiled to address common issues that we have discovered over time such as workflows taking a long time to process, publish and start, as well as errors such as “Failing to start (retrying)”.

There are a number of different factors which can affect the performance and success of workflows, which are fully outlined. This document provides guidance on identifying, responding to and avoiding these issues.

3 Workflow size and performance

As a workflow grows in size and complexity, eventually some common issues will be encountered. There is no hard limit in the size of a workflow, and no definite way of measuring the size. The symptoms outlined in the following section are indicators that a workflow is too large or complex.

NOTE: From our experience, exported workflows whose file size is larger than 500-600KB may start to cause performance issues. However, this is only a general rule; the size of the workflow is not an exact figure and depending on the SharePoint configuration where Nintex Workflow is installed, this value may increase or decrease.

3.1 The workflow takes a long time to start and/or publish

Larger workflows will take longer when publishing or running for the first time. When a workflow is taking longer than a minute to publish, it is the first indication that the workflow is getting large.

You can watch the memory usage rise on the front end web server as the workflow is published to get an indication of the resources being used.

As the workflow gets larger, eventually a situation can be encountered where the application pool in Internet Information Services recycles. When this happens, the user publishing the workflow will often be presented with a login prompt during the publishing operation. Once this occurs, the workflow will not publish and has passed the threshold of what the server can handle.

The resource and time taken to publish is mainly taken up by a call to SharePoint to 'validate' the workflow. If the publish time is more of a nuisance, and the design has been finalized and is known to not cause issues at runtime, the validation step can be skipped to speed up the publish process.

In Nintex Workflow 2007:

1. Open the **Settings** menu of the designer.
2. Choose 'Start up options'.
3. Check the 'Publish without validation' box.
4. Publish the workflow. It will be noticeably faster.

In Nintex Workflow 2013 and Nintex Workflow 2010:

1. In the ribbon, click the **Workflow Settings** button.
2. Scroll down to "Publish without validation" and check the box.
3. Click the **Save** button then publish the workflow.

Please note though that not validating the workflow is not a solution to workflow size issues. A similar process occurs when the workflow starts for the first time (when it is compiled and cached

for future operations). Skipping the validation step may just postpone issues until the workflow starts.

In general, it is better to publish workflows with validation, especially for troubleshooting purposes. If validation is skipped, potential issues will not be found and problems that may occur later at runtime will be much more difficult to diagnose.

> The validation described above is not the end user validation that presents warning icons on workflow actions. It is a lower level validation performed by SharePoint after regular validation has completed.

3.2 Resources used when a workflow starts

When a workflow starts for the first time after being published, it is compiled and then cached for future executions. If the workflow is large, this compilation time can take a significant amount of time and resources on the server.

Usually when a workflow starts it is started in the w3wp.exe process that runs the web site. While it is compiling, the web site may be unresponsive on the server.

3.3 Workflows are failing with specific errors

The specific errors are outlined in the next chapter, but they include:

- Cannot find a corresponding human workflow task ID for this task
- This task is currently locked by a running workflow and cannot be edited.
- Failed to start (retrying)

3.4 Architectural considerations

Nintex Workflow imposes no specific size limitations on workflows because it shares the same architecture as standard SharePoint Designer workflows and as such, shares the same performance factors. Therefore, where a 500 KB workflow file might present no problems in one system, it may in another. This largely depends on the amount of resources the environment has available to it and the load within it. For example, a workflow may have been developed and extensively tested in a staging environment and run without issues then using, yet in a Production environment with greater load and strain on system resources, it may be too big.

Microsoft workflow performance details are described in the following TechNet articles:

http://technet.microsoft.com/en-us/library/gg508755.aspx

SharePoint 2007 - http://msdn.microsoft.com/en-us/library/dd441390%28v=office.12%29.aspx

SharePoint 2010 - http://technet.microsoft.com/en-us/library/cc298801(v=office.14).aspx

SharePoint 2013 - http://technet.microsoft.com/en-us/library/cc298801.aspx

4 Common errors and issues

4.1 Common error messages

4.1.1 Cannot find a corresponding human workflow task ID for this task

You find the error "Cannot find a corresponding human workflow task ID for this task" when clicking a link in a workflow email notification. When the user attempts to view the task, the page queries the Nintex Workflow database for the record linked to the task item. When a record is not found, this error message is displayed.

There are a couple of reasons why the data can be missing from the Nintex Workflow database:

1. Data has been manually purged from the database during a maintenance operation.
2. Database or SharePoint migration has resulted in the site collection accessing the wrong database, or a site collection ID has changed so it does not match the recorded data.
3. A workflow has restarted after timing out while trying to commit at run time.

The third case is relevant to performance and workflow size. If a workflow fails to execute due to being too large, SharePoint will attempt to run it again from the beginning. While the workflow was running, a task notification was sent to a user, a task item was created in SharePoint and records to track the task were recorded in the Nintex Workflow database.

When the workflow restarts, the workflow detects this and clears any existing records from the database so the status tracking can begin again. However, the original task item still exists in SharePoint, so if a user uses a hyperlink from the original notification, they will see a task with no backing data in the database. This will result in the error “Cannot find a corresponding human workflow task ID for this task”.

4.1.2 This task is currently locked by a running workflow and cannot be edited.

While there can be many causes for this error, one of them can be when you have too many tasks in parallel in a single workflow. This often means that multiple processes are attempting to access a workflow task at the same time. A large workflow has more tasks and more updates in the single workflow, so more chance of causing a lock. Also if a lock does occur, a larger workflow can make it difficult to determine the point at which the lock happened while a smaller workflow will always be easier to diagnose. Other causes of this error are explained in the following Nintex Connect article: Task Lock - Task is locked by a running workflow and cannot be edited.

4.1.3 Failed to start (retrying)

On starting, the workflow is unable to successfully compile and execute within the SharePoint system’s configured time-out.

SharePoint throttles the number of workflows executing simultaneously to prevent workflows from using too many system resources. If more than the allowed number of workflows try to execute at the same time, those above the limit will be queued to start later. The status of the queued workflows will display as ‘Failed to start (retrying)’ and eventually they will start, usually after five

minutes. If the workflow is still unable to start the workflow, it is queued to start in 10 minutes. The delay is progressively increased each time the workflow fails to start.

Other times, 'Failed to start (retrying)' indicates a regular error and the workflow will not attempt to restart. To determine more information about this issue the first place to check is the SharePoint logs. The default locations for the SharePoint logs are as follows.

- SharePoint 2013: `c:\program files\common files\microsoft shared\web server extensions\15\Logs`
- SharePoint 2010: `c:\program files\common files\microsoft shared\web server extensions\14\Logs`
- SharePoint 2007: `c:\program files\common files\microsoft shared\web server extensions\12\Logs`

Errors related to this message will be logged with a category of 'Workflow Infrastructure' at the time the workflow attempted to start.

This behaviour is sometimes related to an issue with the SharePoint Timer service (2013, 2010) or Windows SharePoint Service Timer (2007). As such, the first step is always to restart this service on each server within the SharePoint farm. This act is not expected to cancel or error workflows that are already running but will re-make its job queue and freshen it.

4.2 Common issues

4.2.1 Timeout errors when updating an item / when a workflow commits.

When a workflow makes an update to a SharePoint item the changes are added to a batch and processed when the workflow reaches a "commit point". "Commit points" typically occur when the workflow reaches a delay, a task or the end of the workflow. At a commit point, the workflow executes all the work items that were batched in its queue, and SharePoint persists the state of the workflow into the SharePoint content database.

A timeout can occur when the workflow state takes too long to persist, or when the workflow engine takes too long to process the queued operations.

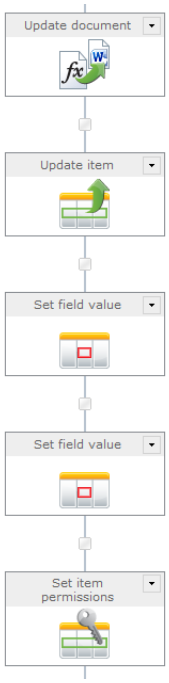
The time it takes for the workflow engine to commit the items is dependent on the number of operations to perform, and the time it takes for SharePoint to perform each individual operation.

For example, the workflow may be attempting to process a large number of 'Update list item' actions at a single commit point. To work around this type of scenario, additional commit points can be inserted to break up the work loads into smaller batches. A 'Commit pending changes" action or a "Delay for" action can be added to force a check point.

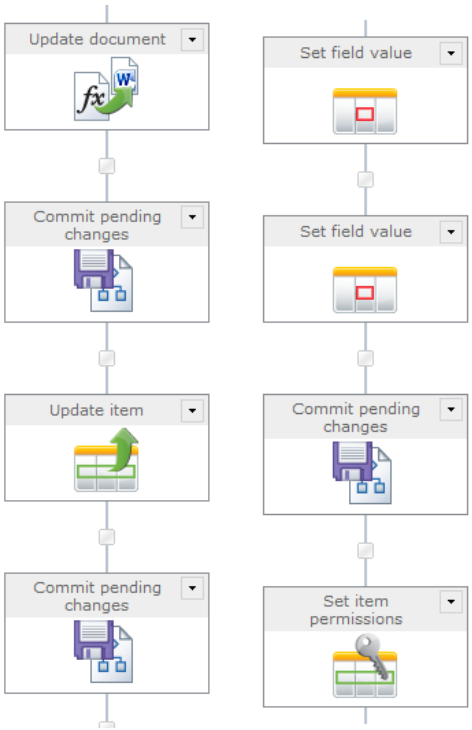
An example of SharePoint taking a longer-than-expected time to perform an operation can occur when other SharePoint performance limitations impact the workflow. For example, if a list contains thousands of items, performing an update or setting permissions on a single one of those items can take a significant amount of time. It is important to remember that performing an operation through

a workflow is affected the same way as the UI or other custom code will be by SharePoint performance.

This will help separate the updates into smaller, much more manageable processes. For example in this simple sequence of workflow actions:



These actions will not actually be committed as they execute: instead they will only finalize the changes when a commit point is reached. This process should be changed to the following:



IMPORTANT NOTE: If a "Commit pending changes" action does not always work, substitute it for a "Delay" or "Pause" action instead. The "Set item permissions" action is one that is known to take longer than others to complete what it's doing, hence a Delay or Pause should always work.

Overall, commits and delays/pauses will ensure the updates are committed as the actions are run and reduce the stress on the workflow processing engine.

4.2.2 Multiple workflows accessing/updating an item at the same time

When designing a workflow process, a common error is to have two workflows starting at the same time, both set to update the same item. This can lead to issues where an update can fail and the workflow will end in an error, usually "Update List Item Failed".

When running a large workflow it can be very difficult to determine which action caused the issue. Also, due to the large amount of actions running, having the correct timing or sequence of update actions is also difficult to manage. To resolve these types of issues splitting the workflow is also recommended.

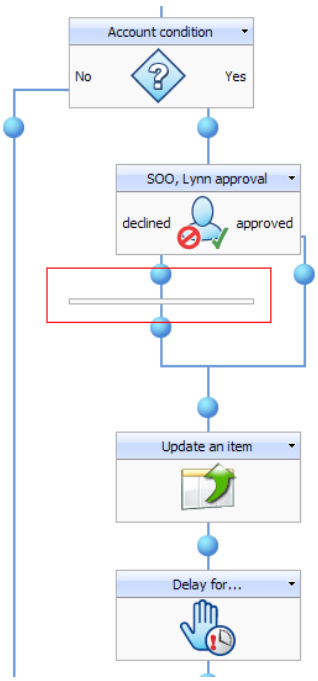
A good method would be to have a set of updates in the first workflow, commit the changes (with a "Commit pending changes" or a "Delay/Pause" action) immediately afterwards then use a "Start workflow" action to initiate the second workflow process. Using this method, both workflows will

essentially start automatically however they will not clash and both attempt to update the item at the same time.

For more information on troubleshooting these types of issues, see the following Nintex Connect article: Designing your Workflow - Commit Pending Changes Action NW2010 & NW2013.

4.2.3 Actions not displaying correctly in the workflow designer UI (Nintex Workflow 2007)

When designing workflow with 'nested' actions, in certain cases an issue may arise when inserting a new action - the action does not display correctly. For example the action will display as follows:



This is an issue with Internet Explorer and 'nested actions'. The deeper you go with more nested actions (for example a "Set a condition" action within a "Set a condition" etc) you may reach a point where the actions are no longer being correctly displayed. This is a limitation within Internet Explorer. The Workflow Designer is built on HTML/ASP.NET technology. Therefore, what you see when looking at and designing a workflow, are HTML elements such as "divs". When you have actions like "Run parallel actions", each parallel action arm in the designer is inside the "div" of its parent.

The problem is the number of nested elements that JavaScript can cope with. In the past when we've seen it, the problem was caused by about twelve consecutive "Set a condition" actions, just by way of example.

To work around this issue you would need to re-design your workflows to avoid the sequential nested actions need or split the workflow process into smaller sub workflows and then send

variables between the workflows using start data. Please see the section in this document entitled "Splitting a workflow into smaller sequential workflows".

5 Key Nintex Workflow design factors in reducing system load

5.1 Task list issues and the amount of approval actions used

Each approval action creates an associated item in the task list it is set to use and many approval-type actions may also be assigned to more than one user. According to Microsoft documentation, more than 2000 list items may cause a performance decline. Some key things to take into account when designing your workflows are how many tasks are being generated by each workflow and how many workflow executions are being run per day/week/month. Any of the “Request approval”, “Request review”, “Assign a flexi-task”, “Assign a To Do task” and “Request data” actions generate tasks. By default, all libraries and lists in a single site use the same default “Workflow tasks” list.

You can pre-empt or mitigate this behaviour by having a separate task list per workflow. All you need to do is create a SharePoint task list, then set the workflow to use it by using the "Start options" in the workflow designer. Simply select the task list you created and once the workflow is published, it will from then on use it. The publishing process will also add all the necessary content types to the new task list.

If you are already well established with your use of workflows and the task list growth is very large, creating a maintenance plan to gradually change the workflows to use different task lists is a viable method to improve performance.

Please also note large task lists can also lead to a ‘task locked’ issue, which is described in more detail in the following forum post: http://connect.nintex.com/forums/thread/6503.aspx (this applies to all versions of Nintex Workflow for SharePoint). Workflow list limitations and performance factors can also be found in the ‘Architectural considerations’ section of this document.

5.2 Certain actions have a heavier processing load

Actions that execute external processes such as the ‘Execute SQL’ or ‘Query LDAP’ action have a greater impact on performance. Splitting the workflow into a separate sub workflow for all these actions may have a positive impact. Actions such as ‘Log to history list” and ‘Build dynamic string’ are very light in terms of processing, and the workflow can contain many of these without affecting performance.

Update type actions should be processed in small batches to ensure they are committing in the correct order, as well as to increase performance and allow an end user to diagnose any potential issues. For more information about committing, please refer to Appendix 1 of this document.

5.3 Delay actions (state machine/loops)

All delay type actions are controlled by the SharePoint timer service. Reducing the number of delay actions within a workflow can increase overall workflow performance and execution speed.

By default the State Machine and Loop actions have an internal hidden delay (5min) to stop potential infinite loops being designed and then taking up all system resources.

Safe looping can be disabled via the Nintex Workflow Central Administration Global settings select “No” for the Enforce Safe Looping option. For it to take effect, you must also perform an IISRESET on all web front-end servers and republish the workflow.

For information on other factors that help increase delay performance and accuracy, see the following Nintex Connect article: Workflows not continuing - Timer Service and Delays.

5.4 Amount of concurrent workflow processes running/workflows starting at the same time

Another key performance factor is the amount of workflows running on a list or within a site at the same time. Microsoft recommendations are mentioned earlier in this document.

Our recommendation is that any new workflow process should first be tested extensively within a staging environment under normal and increased load conditions. Each SharePoint setup will vary and depending on other factors such as the amount of users, other software, latency, etc, these limitations will often not follow directly with Microsoft guidelines. However, the points mentioned in Microsoft’s TechNet article should be a valid starting point on which to base your design process.

5.5 Amount of inaccessible data being logged as workflow history

You can reduce drastically what is being logged in to workflow history and into the dbo.workflowprogress database table by making a few simple changes and republishing the workflow.

First, identify any actions or sub-processes that are repetitive. A Loop action or a "For each" action would be perfect candidates. Edit the workflow then open the configuration screen of one of those repetitive actions. Click the "Common" button in the ribbon and check the box labelled "Hide from workflow status". Click "Save". The workflow action will now no longer add entries to the dbo.workflowprogress database table. To be safe, do the same for any workflow action that is included in the sub-process that makes up the workflow action, so anything contained within the loop or within the "For each" action.

Repeat this process for the rest of the workflow actions that are repetitive ("For Each", "Loop", "Collection Operation") then publish the workflow. New instances of the workflow should now log much less and hence help keep down the size of the workflowprogress database table, maintaining healthier performance.

Please note: the data no longer being logged through a repetitive process was not able to be accessed for historical purposes anyway, making it superfluous.

6 To split or not to split – preliminary considerations before breaking up a large workflow

The whole concept of a workflow being too large is defined by the current state of a user's environment, the hardware available, the version of SharePoint patching installed and SQL Server performance. In most cases Nintex has investigated, when a timeout on publish or start has occurred there have also been other back-end related issues and general SharePoint timeout errors (in the application event log) as well as occasional network/SQL connectivity issues.

As mentioned earlier, this is a dynamic issue that varies between each SharePoint farm. Based on numerous investigations essentially this issue is due to time-outs caused by the amount of time it takes to compile the workflow. When a workflow is first run or published it is 'compiled' and a temporary copy is stored within SharePoint. This process can time-out, as can subsequent attempts to start the workflow.

In many cases, the only option is to redesign the workflow into smaller, connected sub-processes. Dividing a workflow in such a way normally circumvents this issue as each workflow will compile and run in a smaller amount of time and consume fewer resources for a shorter amount of time.

However, that can be a time-consuming process in itself so before going to the effort, you can try increasing some of the SharePoint time-outs that relate to workflow processes.

Please note: User Defined Actions (UDAs) may appear to reduce the amount of work being done when viewed on-screen, but that is not the case. At run-time, all the actions used in the UDA are compiled as part of the workflow.

7 Splitting a workflow process into smaller sequential workflows

If a workflow process has become overly difficult to view, contains multiple approval stages or is running into issues/errors when publishing or initiating (especially time-outs), then the overall design should be split into sub-workflows that are started by a preceding workflow in the chain.

7.1 Advantages of splitting a workflow into separate sub processes

7.1.1 Avoiding the repetition of a long, complex process

In a complex workflow process, an error may occur for an unexpected reason and in these cases the workflow will need to be restarted from the beginning. This is standard SharePoint behavior and is how declarative workflows operate; Nintex cannot alter this behavior.

This can potentially lead to a workflow that was nearing completion but failed during its last set of actions, which in turns means the workflow must be restarted from the very beginning and any approval type actions must be repeated.

If a workflow has been split up into sub workflows, the entire process will not need to be started from the beginning. The sub-workflow that failed can simply be started manually on the item and the appropriate start data also manually entered. This not only saves time but ensures if an unexpected issue does occur it will have the smallest impact possible.

7.1.2 Ease of troubleshooting

Smaller workflows also have the advantage of being much easier to diagnose and troubleshoot. If an error occurs in a workflow with hundreds of actions, the first step may be to reproduce the error. This can be very difficult and often is a process of running the workflow manually and editing it to determine bit by bit which actions are causing the issue. Once the issue has been determined the cause can then be investigated.

A workflow with fewer actions is much easier to manage and to determine which section or action is causing an issue. Also, in terms of error replication, a workflow with fewer actions is advantageous since there are simply fewer factors at play which could stop the real problem from being discovered.

7.1.3 Eliminating time-outs

Each time a workflow is run for the first time, a temporary version is created and compiled on the SharePoint server. When running a workflow that is overly large this process can take some time and create extra load on the server. Smaller workflows with fewer actions will compile faster and increase performance.

7.2 Disadvantages

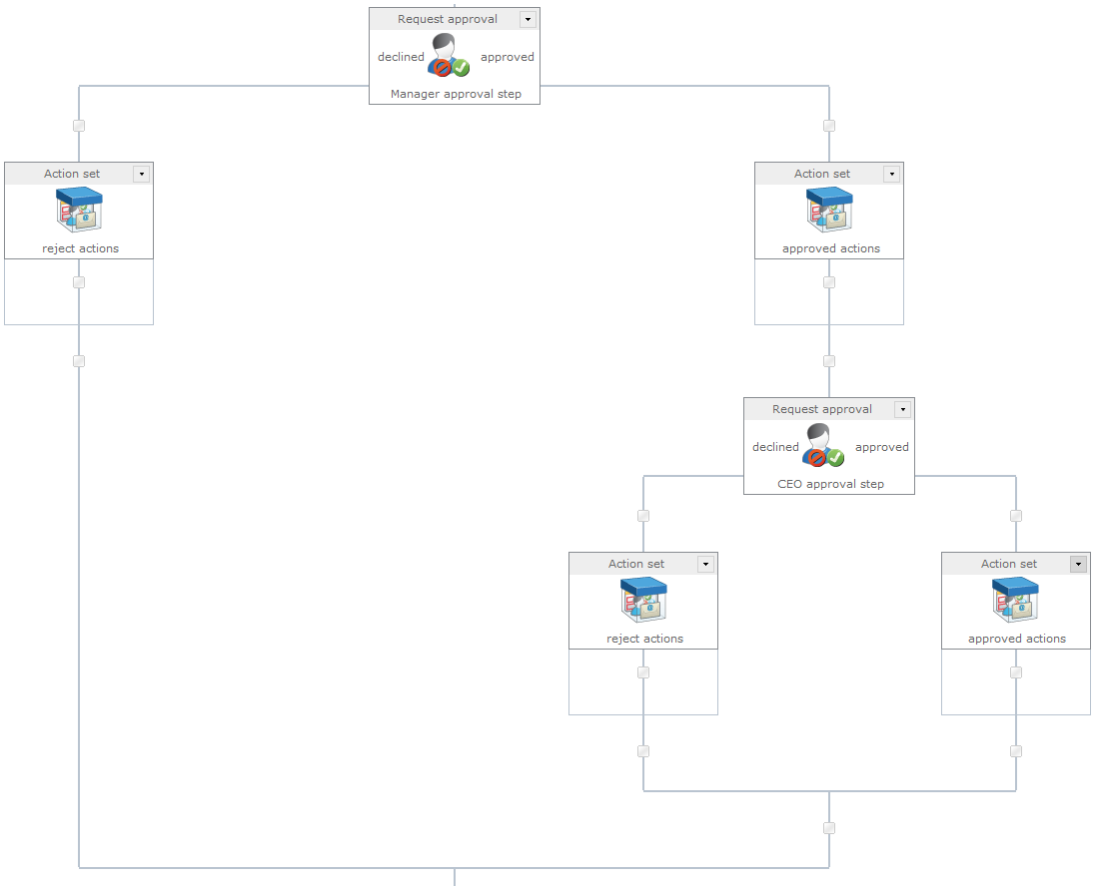
7.2.1 Understanding your process at a glance

Splitting up a workflow means you will no longer be able to see your entire process in the one place.

7.3 Approach to splitting

Given that there could be thousands of combinations and permutations of workflow designs, it's very difficult to describe one way to split one complex workflow into separate parts. Therefore, a general approach is offered by way of a relatively simple example.

For example, this simple workflow has two approval stages:

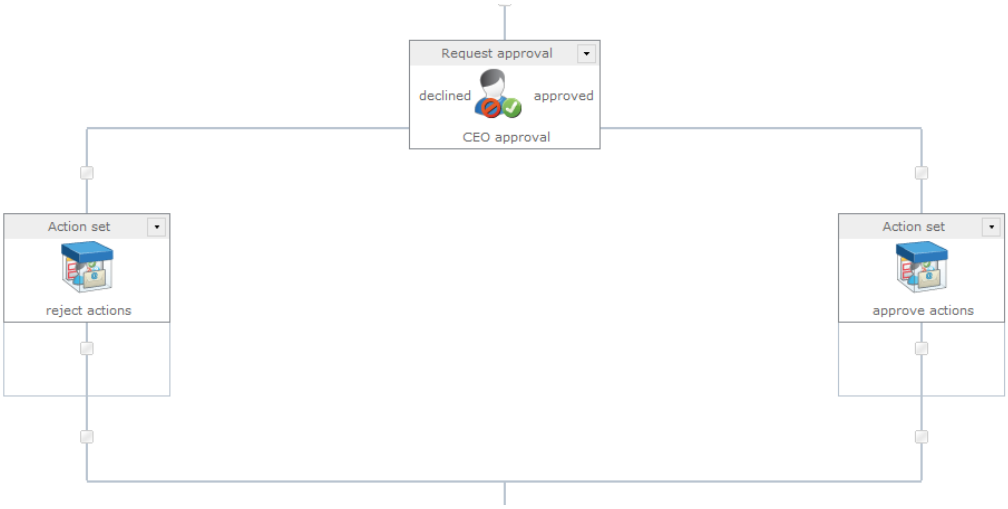


There can be any extra amount of actions however the essential part is that the workflow has two stages, the initial approval and then the second level approval. If this workflow was having issues, the logical step would be to split up the workflow into two workflows; 'stage 1 approval' and 'stage 2 approval'.

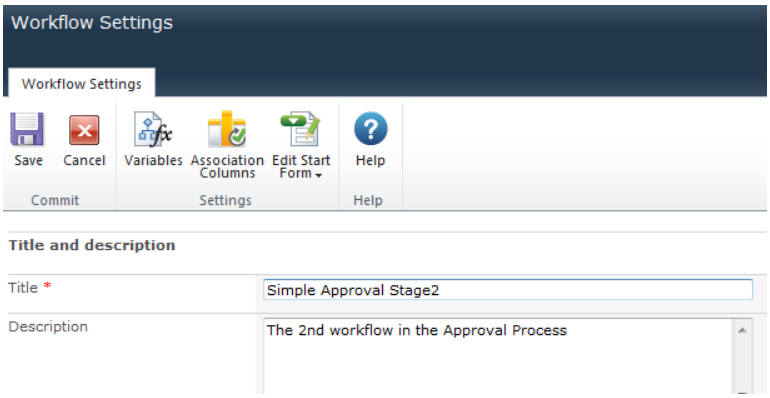
To do this you would first need to determine where the second approval process begins and therefore where best to divide the workflow. In this simple scenario it is clear and the workflow can be divided into two workflows each containing an approval action.

In other cases it might not be as simple; a general rule is that if you are starting a new set of logic actions, approval actions or are able to group a set of actions then this can become a separate workflow.

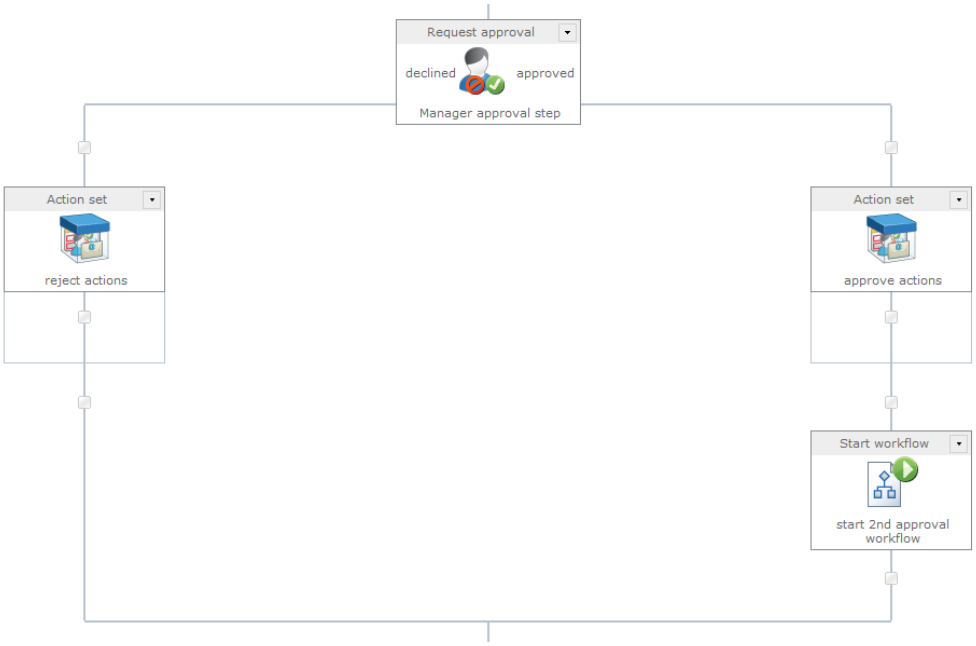
Following on with this example, the initial workflow can be edited and all the irrelevant actions removed.

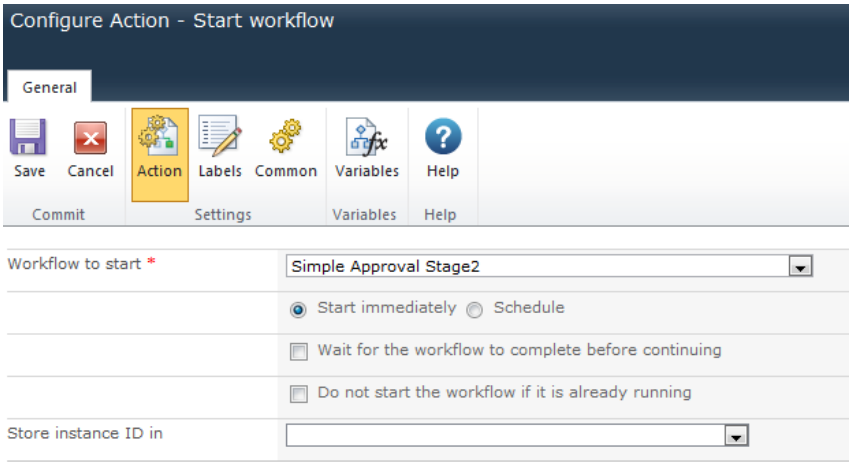


The workflow title can then be changed under workflow settings and the workflow will be published under a new name.



The original workflow can then be edited and the second approval can be replaced with a “Start workflow” action that will start the second workflow in the overall process. This action will be configured to start the “Simple Approval Stage2” workflow.





A particularly powerful method of starting and scheduling workflows is with a web service method. Using this method will also allow you to pass variables between workflows using “Start data”. For more details on how to pass variables between workflows, as well as how to configure the “Call web service” action, see the following Nintex Connect article: Start a Workflow using a Web Service.

8 Appendix 1 – How the "Commit pending changes" action works and when it is best used

Some actions in SharePoint workflow don't execute their workload immediately - instead they batch their job. For example, the "Update list item" action doesn't actually update an item immediately, it waits until the workflow commits. The workflow commits at a delay action, a task action or the end of the workflow. So when the update list item action runs, it just registers that it needs to update the item, the item actually updates on commit.

The SharePoint workflow engine doesn't necessarily commit batched operations in the order they are displayed on the designer. For example, if you had the following actions in this order:

- Set item permissions action (Nintex)
- Update list item action (Microsoft SharePoint)
- Set permissions action (Nintex)

These would actually execute in this order:

- Set permissions action (Nintex)
- Set permissions action (Nintex)
- Update list item action (Microsoft SharePoint)

Why? Firstly, because there are actually two batches, the Microsoft batch and the Nintex batch (any other third party has their own batch). This is because third parties cannot add operations to the Microsoft batch. Secondly, all items in a single batch are executed before actions in another batch**. The batch that is executed first depends on the first activity: If the Microsoft SharePoint action was encountered first, then all the Microsoft actions would run before the Nintex actions.

8.1 The "Commit pending changes" workflow action:

The "Commit pending changes" action is another point where a workflow will execute all its batch operations. So, modifying the above example:

- Set permissions action (Nintex)
- Update list item action (Microsoft SharePoint)
- Commit pending changes
- Set permissions action (Nintex)

In this case everything will run in order. The Nintex batch will run first because the Nintex action is first encountered, but in this scenario there is only one action in this batch. The "Update item" action will run. Then the workflow will commit, and the final "Set item permissions" action is in a new batch.

****IMPORTANT NOTE:** That one batch EXECUTES before another, does not mean that the first batch to execute will FINISH its processing before the second batch executes and finishes. This can cause timing issues. As a general rule:

If you try a "Commit" action and it only works intermittently, replace the "Commit" action with a "Delay for" action (2007) or a "Pause" action (2010).

This will eliminate any risk of timing issues between different batches. A common scenario when this occurs is using a "Set item permissions" action followed by a Commit then an "Update list" action. The "Set item permissions" action takes longer than other actions to finish its task and can end up taking effect AFTER the "Update list" action, causing problems. Replacing the Commit action with a Delay in this scenario is known to fix the timing issue.

8.1.1 Actions that are processed in the SharePoint batch

- Cancel check out
- Check out item
- Copy an item
- Create an item* (hybrid, see below)
- Delete an item
- Log in the History List
- Set field
- Update item
- Wait for item update

8.1.2 Actions that are processed in the Nintex batch

- Check-in item* (hybrid, see below)
- Set approval status
- Set item permissions
- Start workflow
- Stop a workflow
- Update XML

8.1.3 Hybrid actions

- **Check-in item** - if you choose Major check-in and minor versions are not allowed on the library, it's effectively a SharePoint action (and batch). For other options, it's batched by Nintex.
- **Create an item** - some special library types such as wiki pages, have been enhanced by Nintex as well as a number of other options. However it most commonly works as part of the SharePoint batch. It acts as a Nintex action for the following SharePoint objects:
  - Discussions and Replies
  - DocumentSet (Nintex Workflow 2010)
  - WikiDocument
  - Folders

9 Appendix 2 – Checking performance on your server

Here are few things you can do the check the performance on your server.

There is a good introduction on how to use perfmon here: http://technet.microsoft.com/en-us/magazine/2008.08.pulse.aspx?pr=blog.

The specific perfmon stats that are useful in the context of monitoring SharePoint can be found here:

http://blogs.msdn.com/b/joelo/archive/2007/01/16/good-list-of-performance-counters.aspx.

You can also track all of the counters listed under the "Windows Workflow Foundation" category, for both the "w3wp.exe" and "owstimer.exe". Trending on all the counters would be a reasonable approach. This can give you an insight into the number of workflows running on your system at any point in time to look for spikes in server load from the workflow subsystem.

To get an idea on statistics, please run these two queries on your Nintex Workflow database. This query will show the workflow instances that contain lots of records in the Nintex Content database. It will help identify if any workflows are in an infinite loop or not.

```
select I.WorkflowName, I.SiteID, I.WebID, I.ListID, I.ItemID, I.WorkflowInitiator,
COUNT(P.WorkflowProgressID) as ActionCount
from WorkflowInstance I inner join WorkflowProgress P
on I.InstanceID = P.InstanceID
where I.State=2
group by I.WorkflowName, I.SiteID, I.WebID, I.ListID, I.ItemID, I.WorkflowInitiator
order by COUNT(P.WorkflowProgressID) desc
```

This query will show how many workflows are currently running in the environment.

```
Select
P.ActivityID,
A.ActivityName,
COUNT(Distinct I.InstanceID) as WorkflowsInProgress
from workflowinstance I
inner join WorkflowProgress P
on I.InstanceID = P.InstanceID
inner join Activities A
on P.ActivityID = A.ActivityID
where [State]=2
and P.ActivityComplete = 0
and not exists
(select WorkflowProgressID from WorkflowProgress P1 where
P1.InstanceID = P.InstanceID and
```

```
P1.SequenceID = P.SequenceID and
P1.WorkflowProgressID > P.WorkflowProgressID
and P1.ActivityComplete = 1)
group by
P.ActivityID,
A.ActivityName
order by WorkflowsInProgress desc
```