Workflow performance analysis tests and problem reduction

Contents

Introduction

This document is intended to provide some analytical tests that help determine if the SharePoint workflow engine and Nintex databases are being forced to cope with a high amount of workflow related load. References to running through the tests in this document are often made by Nintex Support to determine next steps in improving workflow related performance. They may lead to recommendations such as reducing the size of the WorkflowProgress database table by way of purging data, modifying workflow designs, using more workflow task lists and other such techniques.

If requested, please run the tests below and return the results to Nintex Support.

Recently added were extra steps about identifying problems and mitigating them in the future.

Documents that may be referenced after analysis of these results include:

Reduce the size of the workflow progress database table

Splitting existing SharePoint and Nintex content databases

Workflow performance, timeouts and load reduction. Recommendations.

Nintex Workflow 2010 - NWAdmin.exe Operations

Please note: While the majority of references may be to the 2010 platform, the same concepts apply to the 2007 and 2013 platforms.

Performance tests

TEST 1: Workflows potentially causing bottlenecks

The following query will help find workflow instances that are generating large amounts of database entries. It should be run against the Nintex Workflow content database (or all of them if you have more than one). By default, there is only one Nintex Workflow database (called NW2007DB, NW2010DB or NW2013DB) which combines a configuration and content database.

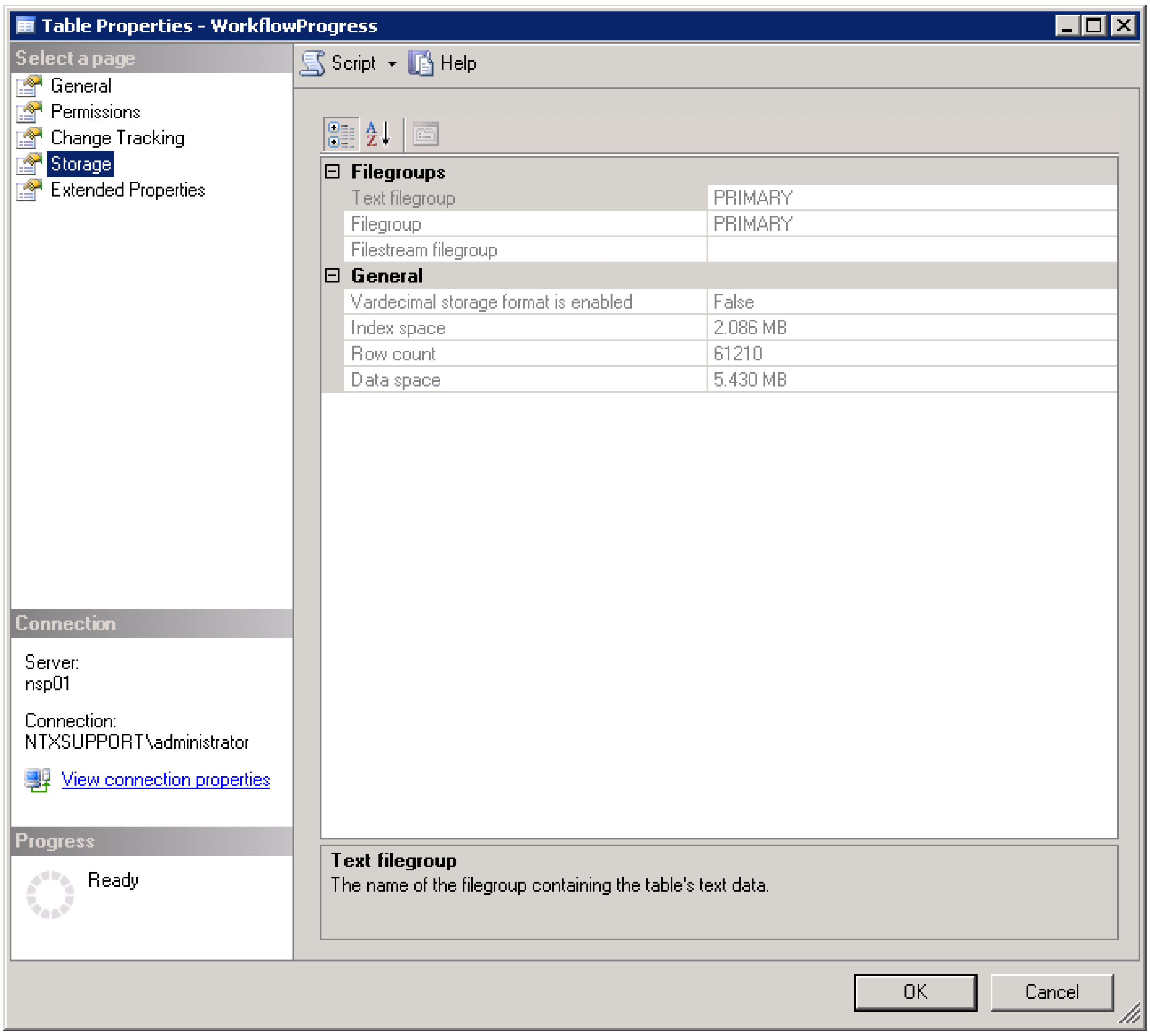
```
Select I.WorkflowName, I.WorkflowInstanceID, I.SiteID, I.WebID, I.ListID, I.ItemID,
I.WorkflowInitiator, I.WorkflowID, I.State, COUNT(P.WorkflowProgressID) as ActionCount
from WorkflowInstance I inner join WorkflowProgress P
on I.InstanceID = P.InstanceID
group by I.WorkflowName, I.WorkflowInstanceID,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

The workflows listed at the top of the results will show workflows generating the most amount of entries. It might indicate that there's a logic flaw causing a bottleneck, the workflow history needs purging or what is logged to the history list needs to be reduced.

The numbers below seen in the "State" column represent the following states:
2 = running, 4 = completed, 8 = cancelled, 64 = errored.

TEST 2: Large workflow history causing performance issues

A large database of workflow history can cause performance issues. To determine how many workflow history records are in the database, open SQL Server Management studio, expand the Nintex Workflow 2010 database, expand the "tables" folder, right-click the "dbo.WorkflowProgress" table and select "Properties". Select the "Storage" page and in the "General" section, note the "Row count" figure.



Server:
nsp01

Connection:
NTXSUPPORT\administrator

Filegroups	
Text filegroup	PRIMARY
Filegroup	PRIMARY
Filestream filegroup	
General	
Vardecimal storage format is enabled	False
Index space	2.086 MB
Row count	61210
Data space	5.430 MB

Text filegroup
The name of the filegroup containing the table's text data.

Depending on the results, using the "NWAdmin" application to reduce the workflow history via the purge commands may be necessary (http://connect.nintex.com/files/folders/technical_and_white_papers_nw2010/entry12004.aspx).

TEST 3: General database size report

We are interested in seeing the amount of data in your database tables which could likely cause the SQL timeout errors.

1. Open SQL Management Studio
2. Expand the Database node
3. Find the Nintex Workflow database (usually named NW2007DB or NW2010DB)
4. Right-click on the Nintex Workflow database and select **Reports** > **Standard Reports** > **Disk Usage by Table**

Please send us a screenshot of the report so we can see which tables are using the most space.

Identifying problems and mitigating them

Analyzing workflow instances for a selected date range

There are two SQL queries below that can be run against the Nintex Workflow database:

1. The first returns a count of workflows grouped by name, site, web and state.
2. The second returns the individual records for the identified problem area (running workflows only).

QUERY 1

The query below returns a count of workflows grouped by name, site, web and state. This is particularly useful if you experience a sudden deterioration in system performance due to an event that causes a very large amount of workflows to start and want to identify where. This works best if you can narrow down a time-frame of when that is likely to have occurred. It will reveal the number of workflows that started between the selected date range per workflow name, site and web ID.

The sections of text highlighted in yellow below represent the date range that can be selected. The format is YYYY-MM-DD HH:MM:SS.SSS where Y=Year, M=month, D=day, H= hour, M=minute, S=seconds. The example below is 01 January 2013 2:30pm to 22 January 2014 9:30am.

```
select COUNT(*) instances,
  WorkflowName,
  SiteID,
  WebID,
  [State]
from dbo.WorkflowInstanceView
where StartTime between '2014-01-01 14:30:00.000' and '2014-01-22
09:30:00.000'
group by WorkflowName,
  SiteID,
  WebID,
  [State]
order by COUNT(*) desc
```

QUERY 2

This second query returns individual records for running workflows (where [State] = 2) between a selected date range. It will return every workflow instance ID which can be used later if you want to terminate a workflow using its workflow instance ID, very useful if you can't otherwise locate which list or site it is in. Similarly you can further filter the information to a more granular level by specifying a site collection GUID, team site GUID, or list GUID.

To query for other running states, alter the where clause (highlighted green) to use one of the following values:

Running = 2
Completed = 4
Cancelled = 8
Errored = 64

The sections of text highlighted in yellow below represent the date range that can be selected. The format is YYYY-MM-DD HH:MM:SS.SSS where Y=Year, M=month, D=day, H= hour, M=minute, S=seconds. The example below is 01 January 2013 2:30pm to 22 January 2014 9:30am.

The areas highlighted in blue are an example of further parameters enabling you to specify a location. If you know it, substitute the GUID of a site collection in the "SiteID" line (and remove the commenting) or the GUID for a team site in the "WebID" line (and remove the commenting). You can reduce it to the list level as well.

```
select InstanceID,
  WorkflowName,
  SiteID,
  WebID,
  ListID,
  ItemID,
  StartTime,
  [State]
from dbo.WorkflowInstanceView
where StartTime between '2014-01-01 14:30:00.000' and '2014-01-22
09:30:00.000'
and  [State] = 2
-- and  [SiteID] = 'ProblemSiteCollectionGUID'
-- and  [WebID] = 'ProblemTeamSiteGUID'
-- and  [ListID] = 'ProblemTeamSiteGUID'

order by WorkflowName,
  cast(SiteID as varchar(128)),
  cast(WebID as varchar(128)),
  cast(ListID as varchar(128))
```

As the site, web and list Urls are not stored in the Nintex Workflow database, you'll need to use PowerShell to determine which Urls are being referenced. Alternatively, you'll need to query the SharePoint content database directly to get this information, as follows:

```
select Id WebId, siteId, FullUrl from dbo.AllWebs

select tp_WebId, tp_ID ListId, tp_Title from dbo.AllLists
```

You'll need to determine the content database against which these should be run.

Terminating specific workflow instances with PowerShell

Armed with a list of problematic workflow instance IDs, you can now use them to begin cleaning things up.

IMPORTANT NOTE: The below information will terminate selected running workflow instances. Please ensure that is acceptable to your business before proceeding. Nintex cannot accept any liability for the decision to proceed with this course of action.

This is especially useful for runaway workflows generating lots of activity in the dbo.workflowprogress table.

FIRST: Run the SQL Query from TEST 2 of this document and retain the output.

FOR LIST AND LIBRARY WORKFLOWS

Open SharePoint Management Console and for the PowerShell commands below, replace references like <Enter SiteID here> with the GUIDs in inverted commas from the above results. Eg: “61543AD-7562-TFR43D-76T5-7645535”

```
#using SiteID column
$site =Get-SPSite <Enter SiteID here>;
$siteurl = $site.url

#Site URL
$web = Get-SPWeb $siteurl;
$web.AllowUnsafeUpdates = $true;

#Get List by guid using listID from DB
$ListID = <Enter ListID GUID here>;
$list = $web.Lists.GetList($ListID,$true);

#use the ItemID from DB ..Note: in Powershell its zero based so itemID
minus 1
$items = $list.Items;
$myItemid = <Enter ItemID here>;
$item = $items[$myItemid - 1]; ## as per the table where the WF ran on, you
can display $item.Title to make sure that you selected the right correct
Item

#Get this Item WorkFlows collection
$WorkflowCollection = $item.Workflows;

#Get WF instance ID from DB
$InstanceWFId = “Insert WorkflowInstance ID GUID here”;
$wf = $WorkflowCollection | where { $_.InstanceId -eq $InstanceWFId}

#Cancel Workflows
[Microsoft.SharePoint.Workflow.SPWorkflowManager]::CancelWorkflow($wf);
$web.Dispose();
```

FOR SITE WORKFLOWS

Open SharePoint Management Console and for the PowerShell commands below, replace prompted, highlighted sections with the relevant value. The first is a URL, the second is the WorkflowInstanceID GUID obtained from TEST 2 in this document. Ensure the GUID is entered in inverted commas: “61543AD-7562-TFR43D-76T5-7645535”

```
$mysite =  Get-SPSite “ENTER YOUR SITE_URL HERE”
$site = Get-SPSite $mysite.id;
$siteurl = $site.url

#Site URL
$web = Get-SPWeb $siteurl;

$web.AllowUnsafeUpdates = $true;

# Site Workflows
    $spWorkflowCollection = $web.Workflows;
    if($spWorkflowCollection)
    {
#Get WF instance ID from DB
$InstanceWFId = "Insert WorkflowInstance ID GUID here"
$workflow = $spWorkflowCollection | where { $_.InstanceId -
eq $InstanceWFId }
      $iWorkflowCount = $spWorkflowCollection.Count;

        if($workflow)
        {
            [Microsoft.SharePoint.Workflow.SPWorkflowManager]::CancelWorkfl
ow($workflow);
        }
    }
$web.Dispose();
```

Reduce the size of the workflow progress database table

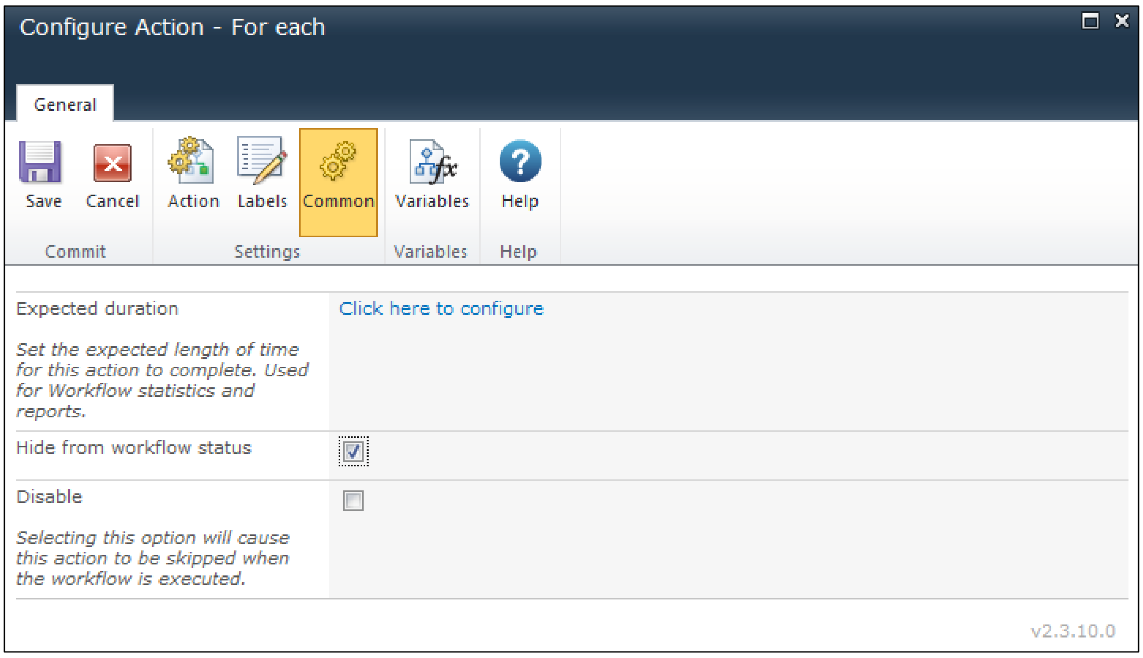
For full detail, refer to this related document: **Reduce the size of the workflow progress database table**

Prevention

Reduce what is logged

If you have workflows that have logic in them that do a lot of looping and processing (including the “Collection” operation), each process will noted in the database, even though only the last result will be logged in the history List. In this sort of situation, it is worthwhile to stop the logging from occurring.

For every action that does programmatic looping or processing, go to the workflow action’s configuration screen, click the “Common” button in the ribbon and select the option “Hide from workflow status”. Actions for which this is helpful are the “Loop”, “For each” and “Collection operation" actions. You should also perform this step for each action contained within these actions. These instructions apply to Nintex Workflow 2010 and 2013.


Configure Action - For each

Expected duration: Click here to configure

Set the expected length of time for this action to complete. Used for Workflow statistics and reports.

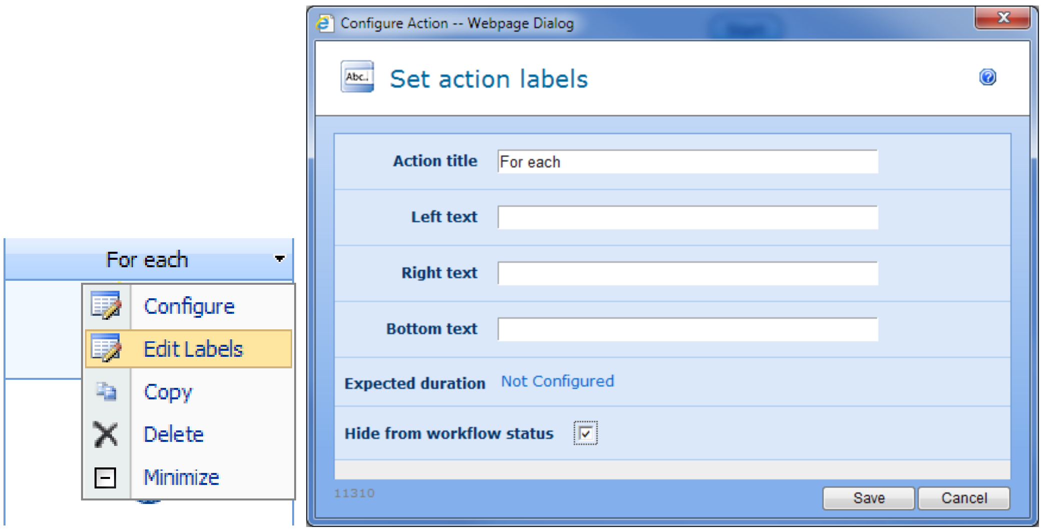
Hide from workflow status: ☑

Disable: ☐

Selecting this option will cause this action to be skipped when the workflow is executed.

v2.3.10.0

For Nintex Workflow 2007, activate the action's drop-down menu, select "Edit labels" then tick the box to "Hide from workflow status".



Configure Action -- Webpage Dialog

Set action labels

Action title	For each
Left text	
Right text	
Bottom text	
Expected duration	Not Configured
Hide from workflow status	☑