Performance improvement:
the workflow progress table

Last updated Friday, June 12, 2015

Legal Notices

For the latest information, please see http://en-us.nintex.com/company/legal.

Contents

Purging workflow data

This document provides information helpful for reducing the size of the workflow progress database table, dbo.WorkflowProgress. This table in the Nintex Workflow content database stores historic data about workflows.

The optimum range for record counts in this table is highly dependent on your SQL infrastructure. Record counts that cause performance issues in some environments do not cause issues in others. We have seen environments with more records than the recommended amount of 10-15 million run without problems. We recommend regularly monitoring your record count and then comparing the record count to your historical results if you start to notice performance issues. If you notice that your record count increased more than expected, it may be time to purge workflow data from the database to improve overall performance of workflows.

For more information on monitoring SQL databases, see the Microsoft article Workflow Scalability and Performance in Windows SharePoint Services 3.0 (Recommendations section).

Purging workflow data reduces the size of this database and helps to optimize performance; however, the purged data is permanently lost. When deciding which workflow data to purge, target deleted sites, workflows with errors, and workflow instances that have created surplus records.

Known issues, tips, and tricks

This section lists considerations for using the purge operation.

- Filters. Because the purge operation results in permanent deletion of workflow data, make sure you specify arguments as filters to choose the category of data that is purged. The filters available are detailed in the "Parameters" section below.
- Timeout. The purge operation has a built-in default timeout of 600 seconds (10 minutes). Purging large amounts of data can cause the operation to reach this timeout. No indication is received until the timeout is reached. To avoid reaching this timeout before completing the operation, either purge smaller chunks of data using filters or increase the timeout using the -timeout parameter.
- Data to target. Data that is commonly considered as least detrimental to historical data integrity follows.
    - Where the workflow state is cancelled or errored
    - Older workflow data (a time and date before which to purge can set)
    - Data for workflows in deleted lists
- Multiple parameters. You can apply multiple parameters to achieve conditions such as "Purge workflow data for workflows that were cancelled or errored before 11pm on 23 June 2012."
- Date format. The date specified must be in the format used by your server. For example, if your server is using US time zone, then specify the date June 23, 2012, as "06/23/2012" (surround the date with inverted commas).

Identifying workflows potentially causing bottlenecks

Use the following query to help find workflow instances that are generating large amounts of database entries. Run the query against all of your Nintex Workflow content databases.

```
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
```

Workflow instances generating the most entries are listed at the top of the results. Very high numbers of entries may be caused by workflow logic that includes a lot of looping and processing, including the Collection operation. You might consider purging data for workflow instances with high numbers of entries.

To purge workflow data for a specific workflow instance, use the -instanceId parameter to specify the instance ID.

Example:

```
NWAdmin.exe -o PurgeWorkflowData -instanceId YourWorkflowInstanceIdGUID
```

where workflowInstanceId is the ID of the instance.

Example with ID:

```
NWAdmin.exe -o PurgeWorkflowData -instanceId 070690D5-D777-4DA9-B577-3E39CB6D4B1B
```

Prevention

If your environment contains workflows with logic that includes a lot of looping and processing, then each process is noted in the database, even though only the last result is logged in the history list. In this situation, you may want to stop the logging from occurring for those actions.

The following workflow actions trigger programmatic looping and processing.

- Collection operation
- For each
- Loop
- Child actions contained within the above actions

To stop logging for a workflow action (Nintex Workflow 2013 or Nintex Workflow 2010)

1. Open the configuration settings dialog box for the action.
2. In the ribbon at the top of configuration settings dialog box, click **Common**.
3. Select the check box **Hide from workflow status**.
4. Click **Save** to save your changes.

To stop logging for a workflow action (Nintex Workflow 2007)

1. On the designer canvas, open the drop-down list for the action and then select **Edit labels**.
2. Select the check box **Hide from workflow status**.
3. Click **Save** to save your changes.

Using PurgeWorkflowData

For usage, parameters, and other documentation for the PurgeWorkflowData operation, search Nintex Connect at community.nintex.com for the NWAdmin guide for your installed version of Nintex Workflow.