

# Maintaining Referential Integrity of SharePoint List Items using Nintex Workflow 2007

---

Author: Mohsin Sheikh, [OBS](#)

SharePoint allows users to create Lookup columns to create a Master/Detail List relationship similar to a relational database - however it doesn't enforce referential integrity checks while deleting an item.

If you delete an item in SharePoint which is being referenced in another list, SharePoint simply deletes the item without any referential integrity checks and the child record is left as a null reference pointing to nothing. In a production environment, this behaviour can cause a lot of issues including skewed, inaccurate data.

This article provides a solution to enforce this functionality by using Nintex Workflow 2007 and a "no-code" solution for easier deployment.

## 1. Disabling the Delete Button

We want to take away the Delete functionality from users since workflows cannot fire on a Delete action without code. The simplest way to achieve this is by creating a custom Permission Access Level based on "Contribute" access and just remove the Delete Item rights. This will also hide the Delete button from List forms.

Name:

Description:

Select the permissions to include in this permission level.  
 **Select All**

**List Permissions**

- Manage Lists - Create and delete lists, add or remove columns in a list, and add or remove public views of a list.
- Override Check Out - Discard or check in a document which is checked out to another user.
- Add Items - Add items to lists, add documents to document libraries, and add Web discussion comments.
- Edit Items - Edit items in lists, edit documents in document libraries, edit Web discussion comments in documents, and customize Web Part Pages in document libraries.
- Delete Items - Delete items from a list, documents from a document library, and Web discussion comments in documents.
- View Items - View items in lists, documents in document libraries, and view Web discussion comments.
- Approve Items - Approve a minor version of a list item or document.
- Open Items - View the source of documents with server-side file handlers.
- View Versions - View past versions of a list item or document.
- Delete Versions - Delete past versions of a list item or document.
- Create Alerts - Create e-mail alerts.
- View Application Pages - View forms, views, and application pages. Enumerate lists.

For the sake of simplicity, we are using a “one to many” relationship here with two lists, “Category” and “SubCategory”. “SubCategory” references the “Category” title column as a lookup field and the objective is to make sure referential integrity is maintained when an item is deleted from the “Category” list (the master list).

## 2. Triggering the workflow on Delete

Since the out of the box delete functionality is gone, we need a way to allow users to submit requests so an item can be deleted. We will achieve this by creating a custom column in the list similar to the screenshot below:

Column name:  
Delete Category

The type of information in this column is:

Single line of text  
 Multiple lines of text  
 Choice (menu to choose from)  
 Number (1, 1.0, 100)  
 Currency (\$, ¥, €)  
 Yes/No (check box)

Description:  
Submit a request for this category to be deleted.

Default value:  
No

We also would want to hide this column from the DispForm.aspx and NewForm.aspx pages. To achieve this, some simple JavaScript can be used to hide the column. This solution should give you a good idea: <http://www.cleverworkarounds.com/2008/03/13/free-mosswws-2007-web-part-hide-controls-via-javascript/>.

### 3. Creating the workflow

Here are the steps to create the workflow which will ensure referential integrity on any child lists.

- **Workflow Variables:** Create three workflow variables based on the given table
  - ChildCount (Number)
  - ChildCollection (Collection)
  - WebserviceURL (Text)
- **Start-up Options:** The workflow should set to fire when existing items are edited. So uncheck the “Start manually” and “Start when items are created” options.
- **“Run-If” workflow action:** Add a “Run-If action” and configure it with the following settings:
  - **Condition:** Compare “Category” (or your list name) field
  - **Where:** Delete Category (or your Yes/No column name we added earlier) equals “Yes”
- **“Query List” workflow action:** Add a “Query list” action below the “Run-If” action and configure it with the following settings.
  - **Editor mode:** Query Builder
  - **List:** <Select your child/detail list here> - in this example it is “SubCategory”
  - **Field:** <Select the Lookup field> in this example it is “Category”
  - **Expand the Filter tab:** Select “Show items only when the following is true:”

- “Show the items when column” <Lookup field> “Category” (in this case) is equal to
  - Insert a reference to the Title field of the master list.
- **Store result in:** ChildCollection
- **Save**

- **Reopen the Query List action:** This time, select the Editor Mode as “CAML Editor”.
  - You will see the CAML generated by the “Query List” action.
  - To make this workflow reusable, we will change the List ID from GUID to the List Name.

```
<Query>
  <Lists>
    <List ID="d244e0e5-595f-47eb-95cf-f11d048f0a45" />
  </Lists>
  <ViewFields>
    <FieldRef Name="Category" />
  </ViewFields>
</Query>
```

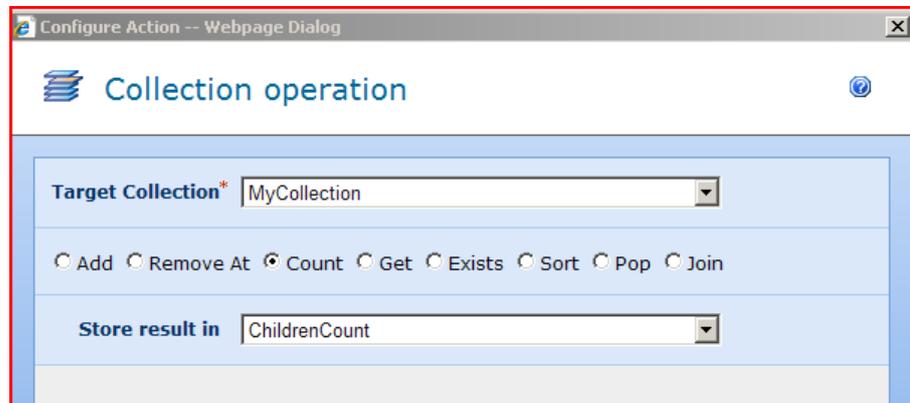
- Change [code block] to

```
<Query>
  <Lists>
    <List ID="Category" />
  </Lists>
  <ViewFields>
    <FieldRef Name="Category" />
  </ViewFields>
</Query>
```

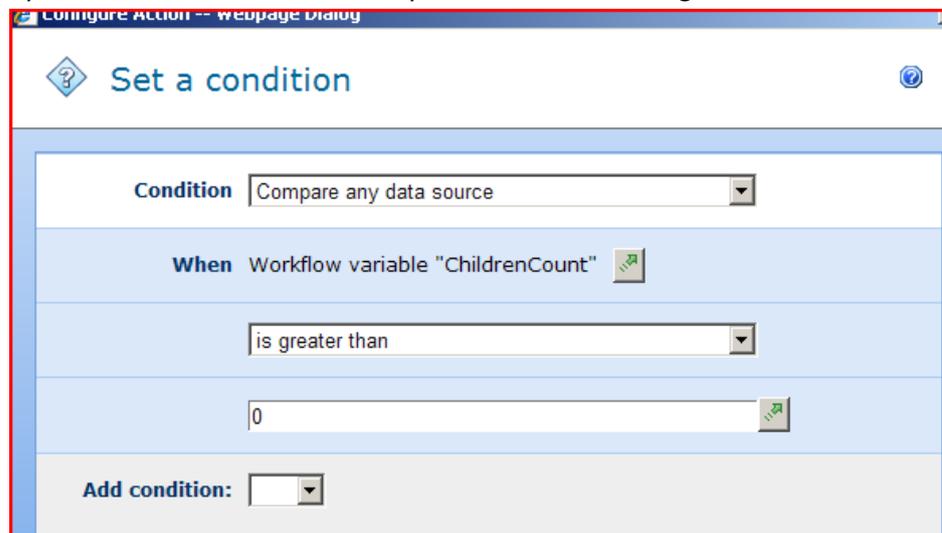
or the name of your List.

- Save

- **Collection Operation activity:** Add a “Collection operation” action and drop it below the “Query List” action. This activity will help us count the number of records returned from the “Query List” action. Configure it using the following settings, readjusting to your environment variables as required:



- **“Set a condition” action:** Add a “Set a condition” action to check if there were any rows returned from the “Query List” action and configure as follows:



- If Yes, use the “Set field value” action to set the field “Delete Category” to No.
- If No
- **Add “Call Webservice” action:**
  - **URL:** <your web server>/\_vti\_bin/list.asmx
  - **Username:** System account or Administrator
  - **Password:** <password of the system account or administrator>
  - The reason we are using impersonation here is since the workflows run in the context of the initiator, and in our case the initiator doesn’t have permission to delete list items, we will have to use impersonation to call a webservice to delete the item.
  - Use the following SOAP Message to delete the item, adjusting to your environment:

*<?xml version="1.0" encoding="utf-8"?>*

```

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <UpdateListItems
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <listName>Category</listName>
      <updates>
        <Batch>
          <Method ID='1' Cmd='Delete'><Field
Name='ID'>{ItemProperty:ID}</Field></Method>
        </Batch>
      </updates>
    </UpdateListItems>
  </soap:Body>
</soap:Envelope>

```

That's it. The webservice call will delete the list item if no child records are present, otherwise the request will be discarded, or you can use a "Request approval" action as required by your scenario.

The complete workflow is on the next page.

