How to find and delete 10,000 obsolete objects using BiToolkits

Recently we have completed an obsolete object cleanup project which identified and deleted over **10,000** unused reports using BiToolkits. This white paper details the challenges we encountered and provides solutions that are readily available in BiToolkits.

Reasons to perform obsolete object cleanup

Over the years, our developers and users have created thousands and thousands reports. To add to the clutter, we often create backup copy of objects before making changes. Those backup copies are also migrated to Production for object comparison and validation (and are often not deleted when done). **Our main motivation to cleanup obsolete objects is due to increasingly difficultly to perform change impact analysis as object dependencies are getting so large**. Of course, improved system performance due to decluttering is an added bonus.

Challenges

There are major challenges in identifying obsolete reports, verifying dependencies, allowing users to review and exclude desired reports from the deletion list, and deleting objects in hierarchical order.

In the past, we have done obsolete object cleanup in a small scale. Using a combination of Enterprise Manager, Ad Hoc SQL access to metadata, and Command Manager scripting, it is quite a challenge trying to delete a few hundred reports, let alone **10,000** reports.

Adding to the complication is incorporating user feedback into the workflow. The users review the report deletion list and can exclude desired reports from deleting. We will show you why this causes major headaches due to object dependencies.

Challenge #1: Identifying Obsolete Reports

First we need to establish reports' obsolescent criteria. In our case, reports that have not changed within the last 6 months and have not run in the last 18 months are considered obsolete. Below are few examples.

- A report was last run 2 years ago and has not updated for years. It is considered obsolete.
- A report was last run 2 years ago but was modified last week. It is **not** considered obsolete.

- A report was created 6 months ago, has not been updated since then, and has never run. It is considered obsolete.
- But if above report is in a web subscription which is scheduled to run at year-end, it should **not** be considered obsolete.
- A report was created 5 years ago and has been not touched ever since. It only runs once a year. It is **not** considered obsolete.

Based on the above obsolescent criteria, we cannot solely depend on Enterprise Manager to identify obsolete reports. Here are the reasons.

- Both report runtime and report modification time need to be taken into consideration.
- If reports have never run, they will not exist in EM.
- If reports were deleted, they will still show up in EM.
- If reports are used as filters, the runtime info is not captured by EM.

Challenge #2: Dependency Check

A report is obsolete only if all its dependents are also obsolete. To illustrate this point, let's assume report A is used as a filter in report B and report B is used as a dataset in document C (A->B->C). If document C or report B does not meet obsolescent criteria, report A cannot be deleted even though it meets obsolescent criteria.

Let's take this illustration further and assume report B is also used as dataset in document D (B->D, A->B->C). If document D does not meet obsolescent criteria, report B and report A cannot be deleted but document C can be deleted if obsolescent criteria is met.

As you can see, the dependency check is quite complex as it has to traverse recursively within nested levels. It is a daunting task if we have to perform dependency checks one object at a time.

Please note, as a result of Dependency Check, dependents of the obsolete objects must be included for deletion if they all meet obsolescent criteria.

Challenge #3: Obsolete Report Exclusion

Allowing users to exclude reports from deletion throws a wrench into the workflow. When a report is excluded from deletion, all the components of that report also need to be excluded from deletion. Using prior example where A->B->C, if C is excluded from deletion, A and B must also excluded even though they meet obsolescent criteria. In our case, users hold off over 1,000 reports from deletion. It is a major challenge try to exclude these reports and all their components from deletion without using a special tool.

Challenge #4: Hierarchical Order Deletion

The last major challenge is to delete objects in hierarchical order to avoid object dependency errors. With over **10,000** objects to delete (including reports/filters/metrics/etc.), it is impossible to organize objects in hierarchical order by hand.

To overcome these challenges, we have incorporated obsolete object cleanup capabilities into BiToolkits. Using BiToolkits, we were able to identify 11,000 obsolete reports in less than 5 minutes. BiToolkits generated 50,000 Command Manager scripts to remove user access to obsolete objects and generated **10,000** Command Manager scripts to delete obsolete objects. Our entire obsolete object cleanup project was done using BiToolkits. Below is the obsolete object cleanup workflow we used.

Steps to Find and Delete Obsolete Objects using BiToolkits

1. **Search for obsolete objects**
   - Use *Obsolete Object Search* option in the *Object Finder* tool to search reports/documents that were last modified prior to 1/1/2017 (replace with your desire date) and last used since 6/2/2017 (replace with your desire date) ➔ name this batch "Obsolete"

   Note, other object types may be returned due to dependencies (such as reports used in filters and filters used in metrics, etc.)

2. **Review obsolete object list**
   - Use the *Web Sub* tool and *NCS* tool to flag reports that are used in subscriptions. Exclude desire reports from the Obsolete list
   - Sort Obsolete list by Folder Path and review reports by folder. Select desire objects to exclude
   - Create the exclusion list by clicking the *Create New Batch* button ➔ name this batch "Exclusion1"

3. **Expand the exclusion list to include components and dependents**
   The Dependents/Components exclusion is needed in an example where Document A has Shortcut S and dataset D. If A is excluded from deletion, S and D must also be excluded from deletion.
   - Use *Components* search tool to find all components (nested levels) on objects in Exclusion1 list ➔ name this batch "Exclusion2" (note Exclusion1 objects are also included in this batch)
   - Use *Dependents* search tool to find Shortcuts (non-nested level) on objects in the Exclusion2 list ➔ name this batch "Exclusion3"
   - Use *Delete Rows* tool to delete all objects except Shortcuts from Exclusion3.

4. **Exclude objects from deletion**
    - Use *Merge Batches* tool to *Exclude* batches “Exclusion2” (Components) and “Exclusion3” (Shortcuts) from batch “Obsolete” ➔ Name this batch “Deletion”

5. **Get user confirmation**
    - Use *Export* tool to export Deletion list to Excel (be sure to include Object IDs). The export can be broken out by user personal folders, share folders, etc.
    - Send Deletion list to object owners and to department leads for delete confirmation
    - Gather user responses and create an object exclusion list (be sure to include Object IDs in the exclusion list)
    - Use *Object ID Search* option in the *Object Finder* tool to bring exclusion list into BiToolkits ➔ name this batch “Exclusion1a”
    - Follow steps #3 and #4 to exclude objects from Deletion list

6. **Web Subscriptions and NCS Final Check**
    - During the period of waiting for user confirmation, new subscriptions might already be created on reports that are in the Deletion list
    - Use *Web Sub* tool to find Web Subscriptions that are using reports in the Deletion list
    - Use *NCS* tool to find NCS Subscriptions that are using reports in the Deletion list. (Note this process might take a while to run as matching between NCS services and obsolete reports are based on report names. If report attachments in NCS are named differently, BiToolkits will not find the matches)
    - Note: Web Subscriptions will be automatically deleted when reports are deleted; however, NCS subscriptions must be manually deleted

7. **Hide obsolete objects**
    Note this step is not necessary but recommended.
    - Use *Change Access Rights* tool under *Command Manager Action* tab to remove user access to objects in the Deletion list
    - Wait a month or so for user feedbacks
    - If users complain of missing objects, restore their access to those hidden objects. We need to log the restored objects as they need to be excluded from the Deletion list

8. **Delete obsolete objects**
    - If there are objects need to be excluded from deletion, we need to repeat step #3 to #6 to exclude objects and their components from the Deletion list
    - Use *Dependents* search (recursive) tool to refresh the Deletion list. If there are new dependent objects that were recently created, we need to add them to the Exclusion list and repeat steps #3 to #4 to exclude them from deletion
    - Use *Delete Objects* tool under *Command Manager Action* tab to generate Command Manager Scripts and to run the generated scripts that delete objects in MicroStrategy

    Note deletion scripts are generated in hierarchical order. There are certain object types cannot be scripted to delete using Command Manager (such as Consolidation, Prompt, Template, etc.) This is a limitation of Command Manager.

9. **Delete Verification**
    - After object deletion scripts are run, use *Create New Batch* tool in *Batch Manipulation* tab to refresh (query against MicroStrategy metadata) the Deletion list. The Deletion list should now be empty if there were no errors when running the deletion scripts