# Adobe® LiveCycle®: Purging processes and jobs

Process data that is generated when a long-lived process is invoked can become too large, resulting in lower LiveCycle ES2 performance and the use of unnecessary disk space. It is good practice to purge process data when records are no longer necessary.

## Purging

When your database approaches capacity, one option is to purge the COMPLETED and TERMINATED processes and jobs. Two purge facilities are available:

- Process purge—Purges rows from the TB_PROCESS_INSTANCE table.
- Job purge—Purges rows from the TB_JOB_INSTANCE table.

### Process purges

In both LiveCycle ES and LiveCycle ES2, you purge processes using the command line facility located in: "c:\Adobe\Adobe LiveCycleES2\LiveCycle_ES_SDK\misc\Foundation\ProcessPurgeTool\ProcessPurge.bat". For UNIX systems, the utility is called ProcessPurge.sh.

#### Usage

ProcessPurge process <administratorUserName> <administratorPassword> <process_name>
<version e.g. * | 1.* | 1.2> <age e.g. 366d | 12h | 3600s>
[ <expression e.g. approved=true>]
[ completed_only|terminated_only|completed_or_terminated ]
[ include_children ]

#### Examples

Purge processes more than 60 seconds old

ProcessPurge process "http://localhost:9080" administrator password
"MyApplication/Processes/MyProcessName" "*" 60s "*" terminated_only include_children

Purge all processes with the fully qualified name "MyAppliction/Processes/MyProcessName," any version, older than 60 seconds, no filter, only status of terminated, including children. Note that 60s can also be interpreted as "keep all processes created within the last 60s."

Purge purges more than 12 hours old

ProcessPurge process "http://localhost:9080" administrator password
"MyApplication/Processes/MyProcessName" "*" 12h "*" completed_or_terminated in-clude_children

In this example, the time period specifies that processes more than 12 hours old will be considered for removal.

### Database queries

You can see the effects of the process purge operation in the LiveCycle database. You can use the following queries to assess the progress of the purge operation. Prior to running the purge, you can use queries based on the following pattern to assess the number of process instances that will be affected:

SELECT ID, SERVICE_NAME, CREATE_TIME, STATUS from TB_PROCESS_INSTANCE

SERVICE_NAME: should match "MyApplication/Processes/MyProcessName"

CREATE_TIME: should be older than 60 seconds or 12 hours in the previous samples to be eligible for purge

STATUS: Terminated = 4, Completed = 2

An example query combining these elements might resemble:

```
SELECT COUNT(*) FROM TB_PROCESS_INSTANCE
        WHERE SERVICE_NAME = "MyApplication/Processes/MyProcessName"
        AND STATUS = 4;
```

## Job purges

### LiveCycle ES job purges

Appendix A contains sample source for building a jar that can be installed in Adobe LiveCycle Workbench ES.

1. Create JobInstanceCleaner.jar (see Appendix A).
2. Install the jar file in LiveCycle Workbench ES.
    a. View Components.
    b. Components > popup menu > Install > Pick the jar.
    c. Select JobInstanceCleaner > popup menu > Start.
3. Create the service JobInstanceCleanerInvoker.
4. Add activity JobInstanceCleaner.
5. Edit the properties of the activity to specify the jobs before which you want to purge.
   For example, 1 day means everything within the last day is kept, and everything before the last day is purged.
6. Invoke JobInstanceCleanerInvoker.

### LiveCycle ES2 SP1 job purges

Use the new health monitor interface added to the LiveCycle Administration Console for SP1:

LiveCycle Administration Console (Top Right) > Health Monitor > Purge records from the Job Manager database

Two options are available:

- One-time—Single purge, executes once.
  Note the "Purge Competed records filter": When this value is set to 0, all jobs that are completed, terminated, or have failed are eligible for purging. This is useful for quick testing of the purge in isolation.
- Automatic purge—Schedules a purge at the schedule you provide.

#### Database queries

You can see the effects of the job purge operation in the LiveCycle database. You can use the following queries to assess the progress of the purge operation.

SELECT ID, SERVICE_NAME, CREATE_TIME, STATUS from TB_JOB_INSTANCE

SERVICE_NAME: should match "MyApplication/Processes/MyProcessName"

CREATE_TIME: should match your specifications from the Purge Competed records filter

STATUS: Running = 2, Completed = 3, Terminated = 5

An example query combining these elements might resemble:

```
SELECT COUNT(*) FROM TB_JOB_INSTANCE
        WHERE SERVICE_NAME = "MyApplication/Processes/MyProcessName"
        AND STATUS = 5;
```

### Process and job purge statistics

The purge functions operate asynchronously, and there is no interference with normal LiveCycle operations. Because of this design, the purge operation might take some time to complete. Tests performed by Adobe on typical application environments yielded purge times shown in the table below. Testing was done in WebSphere 7.0.0.5 on Windows Server® 2003 R2, Enterprise Edition x64 and Oracle 11g on Windows Server 2003 R2, Enterprise Edition x64.

| Table name | Number of rows purged | Purge time |
|---|---|---|
| TB_PROCESS_INSTANCE | 42,000 | 2 hours |
| TB_JOB_INSTANCE | 84,000 | 2 hours |

## After purging

### Global document storage

Jobs and processes might use Intelligent Document Platform (IDP) document objects, managed by the LiveCycle Document Manager, to store forms or documents that were used as process variables, arguments, or results in the process or job. These document objects also reside in the database. The document instances are purged from FileSystem, TB_DM_DELETION, TB_DM_SESSION_REFERENCE, and TB_DM_CHUNK, depending on whether your global document storage (GDS) is on the file system or database.

Removing document data from the GDS is not strictly a data purge operation. GDS documents are created and removed through the course of many LiveCycle activities not related to purging. When jobs or processes are purged, GDS contents related to those jobs or processes might be deleted. From a data purge perspective, the removal of related unused documents is interesting, because a good deal of the database and file system storage related to a process will actually be in GDS document objects referred to by those process instances.

#### GDS on a file system

In this case, remove requests create files under the GDS or backup directory.

The remove request files are examined at every document sweep interval (30 seconds by default plus the time that each sweep takes), and the relevant file lifetime markers are removed per the requests found.

On the next document sweep, any content files found to have no lifetime markers left are deleted.

#### GDS in the database (LiveCycle ES2 only)

In this case, the document data is stored in the tables TB_DM_DELETION, TB_DM_SESSION_REFERENCE, and TB_DM_CHUNK.

Document removal requests are inserted in the TB_DM_DELETION table. The TB_DM_DELETION table is examined at every document sweep interval (30 seconds by default plus the time that each sweep takes). The TB_DM_DELETION table is correlated with the TB_DM_SESSION_REFERENCE table to determine which references are affected and which TB_DM_CHUNK rows can be deleted.

When a TB_DM_SESSION_REFERENCE is the subject of a deletion request, it is first changed to a liveTemporary reference. Up to 50 references can be updated per transaction. Each liveTemporary reference creates a file in the GDS that is removed when the GDS has no lifetime markers for the related content. At the same time that the file is removed, the corresponding session reference is scheduled for deletion.

#### GDS purge statistics

As with the main job and process purges, document removal operates asynchronously and can take some time to complete. By using SQL queries to measure the row counts of the above-mentioned tables, you can monitor the progress of the purge operation. Tests performed by Adobe on typical application environments yielded purge times shown in the table below. Testing was done in WebSphere 7.0.0.5 on Windows Server 2003 R2, Enterprise Edition x64 and Oracle 11g on Windows Server 2003 R2, Enterprise Edition x64.

| Table name | Number of rows purged | Purge time |
|---|---|---|
| TB_DM_CHUNK | 123,000 | 8 hours |
| TB_DM_DELETION | 324,000 | 8 hours |

## Additional information

- LiveCycle ES Database and GDS Backup and Recovery: www.adobe.com/support/livecycle/ts/documents/kb403016/LCES_db_gds_backup_recovery.pdf
- Backup and Restore: http://help.adobe.com/en_US/livecycle/9.0/adminHelp/admin.htm?content=000433.html
- Changing the Archival Mode: http://download.oracle.com/docs/cd/E11882_01/server.112/e10595/archredo003.htm#i1006246
- Purge help: http://help.adobe.com/en_US/livecycle/9.0/adminHelp/admin.htm?content=000799.html#1561423
- Jayan Kandathil's Blog on Purging: http://blogs.adobe.com/livecycle/2010/05/clearing_old_business_processe.html

## Appendix A

### Creating JobInstanceCleaner.jar

To create the JobInstanceCleaner.jar file, compile the Java™ from 5.2 below using these libraries:

- adobe-pof.jar, which you can get out of adobe-livecycle-jboss.ear or the ear for your app server (adobe-livecycle-<appserver>.ear)
- adobe-livecycle-client.jar, which is part of the shipped SDK (for example, C:\Adobe\LiveCycle8.2\LiveCycle_ES_SDK\client-libs\common)

## JobInstanceCleaner.java

```
package com.adobe.custeng;

import javax.naming.InitialContext;
import java.util.*;
import java.lang.Integer;
import com.adobe.idp.Context;
import com.adobe.idp.um.api.UMException;
import com.adobe.idp.um.api.UMLocalUtils;
import com.adobe.pof.GenericObject;
import com.adobe.pof.omapi.*;
import com.adobe.idp.dsc.InvocationRequest;
import com.adobe.idp.dsc.InvocationResponse;
import com.adobe.idp.dsc.clientsdk.ServiceClient;
import com.adobe.idp.dsc.clientsdk.ServiceClientFactory;
import com.adobe.idp.jobmanager.client.*;
import com.adobe.idp.jobmanager.common.JobId;

public class JobInstanceCleaner {

        private static final String POF_OBJECT_MANAGER = "java:comp/env/ejb/POFObjectManagerLocal";
        private static final String BOI_OBJECT_TYPE = "job_instance";
        private static final String POF_DOMAIN = "jobmanager";
        public static final String A_ID = "id";
        public static final String A_STATUS = "status";
        public static final short JOB_STATUS_COMPLETED = 3;
        public static final short JOB_STATUS_FAILED = 4;
        public static final short JOB_STATUS_TERMINATED = 5;
        public static final String A_PUBLIC_ID = "public_id";
        public static final String A_UPDATE_TIME = "update_time";

        public void cleanCompletedJobs(Integer days, Integer hours, Integer batchSize) throws Excep-tion {
                boolean loop = false;
                do {
                        ServiceClientFactory fact = ServiceClientFactory.createInstance(getSystemCont
                        ext());
                        ServiceClient sc = fact.getServiceClient();
                        Map<String, Integer> m = new HashMap<String, Integer>();
                        m.put("days", days);
                        m.put("hours", hours);
                        m.put("batchSize", batchSize);
                        InvocationRequest req = fact.createInvocationRequest("JobInstanceCleaner",
                        "Clean-Batch", m, true);
                        InvocationResponse resp = sc.invoke(req);
                        loop = ((Boolean) resp.getOutputParameter("loop")).booleanValue();
                } while (loop == true);             }

        public boolean cleanBatch(Integer days, Integer hours, Integer batchSize) throws Exception {
                POFObjectSet objSet = getCompletedJobs(days.intValue(), hours.intValue(), batch-Size.
                intValue());
                JobManager jobMan = new JobMan-ager(ServiceClientFactory.createInstance(getSystemCo
                ntext()));
                while (objSet.next() == true){
```

```
                GenericObject obj = objSet.getObject();
                JobId jobID = new JobId(obj.getValue(A_PUBLIC_ID).getStringValue());
                jobMan.disposeJob(jobID);
        }

        int bSize = batchSize.intValue();
        if (bSize > 0) {
                return objSet.size() == bSize;
        } else {
                return false;
        }       }

private POFObjectSet getCompletedJobs(int days, int hours, int batchSize)throws Exception {

        POFObjectManagerLocalHome objManHome = (POFObjectManagerLocalHome) new
                Initial-Context().lookup(POF_OBJECT_MANAGER);
        POFObjectManagerLocalEJBAdapter objMan = new POFObjectManagerLocalE-
                JBAdapter(objManHome.create());
        objMan.setContext(getSystemContext());
        objMan.setDomain(POF_DOMAIN);

        //want to make sure there is at least 1 hour specified
        if (days == 0 && hours == 0) {
                hours = 1;
        }

        POFQuery query = objMan.newQuery(BOI_OBJECT_TYPE);
        query.addFilter(A_STATUS, POFFilter.EQUALS, JOB_STATUS_COMPLETED);
        query.addFilter(A_STATUS, POFFilter.EQUALS, JOB_STATUS_FAILED, POFFilter.OR);
        query.addFilter(A_STATUS, POFFilter.EQUALS, JOB_STATUS_TERMINATED, POFFil-ter.OR);
        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DAY_OF_MONTH, days * -1);
        cal.add(Calendar.HOUR_OF_DAY, hours * -1);
        Date d = cal.getTime();
        query.addFilter(A_UPDATE_TIME, POFFilter.LESS_THAN, d);
        query.projectAttribute(A_ID);
        query.projectAttribute(A_PUBLIC_ID);

        if (batchSize > 0) {
                return objMan.retrieveObjectSet(query, 0, batchSize);
        } else {
                return objMan.retrieveObjectSet(query);
        }       }

private Context getSystemContext() throws UMException {
        return UMLocalUtils.getSystemContext();
}       }
```

**We welcome your ideas and comments. Please send any feedback on this technical guide or suggestions for other topics to:**
techguides@adobe.com

**For more information and additional product details:**
www.adobe.com/devnet/livecycle/

**Adobe Systems Incorporated**
345 Park Avenue
San Jose, CA 95110-2704
USA
*www.adobe.com*