# ERS 10.11 Archive and Purge Design

## Functional Design

This feature will allow ERS functional owners to archive specific reporting periods using an offline batch job. The archive process will preserve all versions of each report in the given period, and the payroll detail for the most recent report. It will also capture all comments associated with each report version.

Base ERS will recommend that ERS operates take a database backup before archiving a period to provide a fallback recovery strategy. The online application must be disabled during an archive batch run.

The application will enforce the following business rules before attempting to archive any reports in the specified period:

* The period must exist
* The period must have Complete (code=C) status or Archive in Progress status
* The period must have only one version marked as current for each report in the period.

If these conditions are met, the application will set the period’s status to a new status, Archive in Progress. If any fail, the batch job will abend with an error message notifying which business rule did not pass.

As part of the archiving process, the application will store index information to support search of archived reports by:

* Period name
* Period ID
* Employee Name
* Employee ID
* FAU
* Account Org ID
* Account Org Name
* Fund Org ID
* Fund Org Name

It will delete data from all online transactional tables related to the reports, including:

* The report header
* All related versions
* All related sponsored project line items
* All related other and non-sponsored line items
* All related report funding source records
* All related Effort Report Resource records

Upon successful completion of the archive process, the application will set the reporting period to a new “Archived” status. Periods with archive status will be listed in the system admin reporting period view, but will be filtered from the online application, including the report list dropdown period filter, and the compliance report drop down filter.

During processing, the application will fail if it encounters any exceptional conditions that prevent it from completing an archive action on a report. The period’s status will remain in Archive in Progress, and will leave the reports archived up to the failure point in the database. If the exceptional condition is something that can be readily fixed, the archive job will be restartable. If the condition is not recoverable, then the recommended database backup can be used to restore the ERS instance to its pre-archive state.

This feature will add a new system administration view, accessible from the existing sys admin view, where authorized users can search for archived reports, select those reports they wish to see, and then produce PDF versions of the archived reports. There will be controls for all indexed fields listed above. The results will be presented in a table format, displaying the period ID, period name, employee ID, and employee name. The results will be paged, with user options to display [series] results per page. The veiw will also present paging controls that navigate forward one page, back one page, forward to the last page, and back to the first page. The table will also present a column of checkboxes, which the user will use to indicate which reports to produce as PDFs. There will be a “select all” button on each page as well.

The PDFs will appear in a separate popup window, to replicate the existing compliance reports user experience. They will be ordered by period ID, employee name, and, within each set of report versions, by the concatenation of the major and minor versions, descending.

This feature will also include a separate purge off line batch job that will delete all data from the archive tables for a given period. Only reports with Archived status will be eligible for purge (the application will enforce a business rule that will abort the purge process if the specified period does not have the correct status.)

## General Technical Approach

When a reporting period is archived, the process will find all report headers with status not equal to ‘Z’ (not generated), produce data transfer objects (DTO), one for every version of the report, and one for the related payroll detail of the final version only, compress the results, and store them in a new archived report table. The application will add indexing for the archived reports. Indexed data will include:

* Employee Name
* Employee ID
* Period Name
* FAU
* Account Org ID
* Account Org Name
* Fund Org ID
* Fund Org Name

There will be a new admin view that will allow users to search for the archived reports they wish to retrieve. When the app executes a query, it will present a list of matching reports (reporting period and employee id). The user can then indicate which ones to download as PDFs. On retrieval, the application will load the set (all reports with the same aer\_cd) of related reports the compressed DTOs , uncompress them, order by the version number descending, and pass on to the jasperPrint report filler and then the PDF exporter. While processing the first (most recent report), the application will also decompress the payroll detail DTO and pass that to the jasperPrint report filler.

The purge batch program will delete all items from the archived report and archive funding source tables.

The report period rows for archived and purged records will be retained. The report period will have a new data status attribute that indicates if it is live, archived, or purged.

## Database design/changes

The application will require a new table to store the compressed report DTO objects, along with the single-value index attributes, employee ID, employee name, period ID, period name:

**ERSArchivedReport**

rpt\_seq\_nbr int primary key

version\_num decimal not null

aer\_cd int not null

rpt\_print\_obj BLOB not null

payroll\_print\_obj BLOB

per\_id int not null

per\_name varchar(30) not null

emp\_id varchar(10) not null

emp\_name varchar(26) not null

The rpt\_seq\_nbr will be the ID of each effort report related to the given aer\_cd. This table will not enforce RI with ERSEffortRpt, since this value will be purged from that table after successfully archiving, nor will it enforce RI with ERSActualER on aer\_cd either, which will also be purged.

This table will require an index on three of the report scalar attribute columns:

* Period ID
* Employee ID
* Employee Name

Period name is only needed for presentation, it will not be searchable.

The application will also require separate index table for associated vector attributes, which include the FAU, the account org ID and name, and the fund org ID and name:

**ERSAchivedFS**

rpt\_seq\_nbr int primary key

fs\_id int primary key

ern\_fau char(30)

acct\_org\_id varchar(6)

acct\_org\_name varchar(50)

fund\_org\_id varchar(6)

fund\_org\_name varchar(50)

This table will require an index as well, on the FAU, the two Org IDs, and the two Org Names. There will be no foreign key relationship to the archive table.

### Deliverables

1. DDL to define new tables, indexes, and alter table on ERSRptPeriod to add new column, added to the three CREATE\_ scripts, and the three v10.11 UPGRADE scripts.

## HIbernate Mappings and Domain Classes

The application will require hbm files for the new tables. The new hbm files need to be added to the applications mappings.xml file.

The general strategy in ERS is to write the hbm file, leveraging off an existing hbm of similar structure, then running the Hibernate Synchronizer to generate the domain and DAO classes. The archive table should have classname ArchiveReport, and the funding source index table ArchiveReportFundingSource. The ID of the ArchiveReport class will be the report sequence number, the ID of the funding source index table will be a compound of the report sequence number and the funding source ID. The funding sources should be mapped into the ArchiveReport class as a one-to-many set. The inverse mapping in the index class will not be necessary.

### Deliverables

1. Two new hbm mapping files for the archive table and the funding source index table.
2. Domain and DAO classes, generated from these mapping files
3. Alteration of the mapping.xml file to include the two new hbm files.

## Archive program

This will be an offline batch program that will take the target reporting period as its only parameter. The archive logic will create the compressed report object by creating a DTO will all report field data in it, and a DTO containing that report version’s payroll detail, compress the DTOs, persist in the archive table, move the indexed resource types to the new archived resource table, and then delete the report and its associated data from the following tables:

* ERSCatItem
* ERSRptFundItem
* ERSEarningsVersion
* ERSEarning
* ERSEffortRpt
* ERSRptFundingSources
* ERSActualER
* ERSERResource

If possible, the report DTO should just be the existing edu.ucop.ers.editcert.ReportStructure class, and its associated component classes, and the payroll detail should use the existing edu.ucop.ers.editcert.DrillDownStructure as its DTO. The new archive logic can use the existing ReportBuilder and DrillDownBuilder classes to construct the DTOs.

The application will require three new report period status indicators:

X Archive in progress

A Archived

P Purged

These need to be defined in ERSConstants, alongside the existing period constants, which start at around line 368 in the current version of this class.

The application must enforce a business rule that only periods with “Completed” status (C) or Archive in Progress status can be archived. At the end of a success run, the application will have to update the status of the reporting period to Archived.

The units of work to be completed by the business logic class include:

1. Load all report headers for a period
2. For each report header that has at least one effort report, create a DTO for each related report
3. For each DTO, compress, instantiate a new ArchivedReport object, populating its fields with the compressed DTO report bytes, employee ID and name, period ID and name, aerCd, and a concatenation of the major and minor versions, cast into a float, then persist the object. For the last (most current) effort report, also capture the payroll detail of the report.
4. Create a new ArchiveReportFundingSource object for each related funding source, populate its fields, and persist.
5. Delete from tables listed above.

Each unit of work should be contained in a protected method, to ease unit testing.

This program will require a new java class that will contain all the business logic described above. It should be included in the edu.ucop.ers.generate package. The program will also require a new driver class with a main method, which performs parameter validation, hibernate initialization, and invocation of business logic program.

The program logic should commit batches in the range of 50-100 records. If possible, the applicaiton should be profiled to determine an optimal size. The general strategy is to maintain a batch count and a total reprots processed count, and at end of an iteration of the main business logic loop, check the batch count. If the predefined batch size is hit, the application can invoke:

if (commitCount > BATCH\_SIZE) {

\_RootDAO.commitTransaction();

\_RootDAO.flush();

\_RootDAO.clear();

\_RootDAO.beginTransaction();

logger.info( totalCount + “Process so far…” );

commitCount = 0;

### Error Handling

All service classes must include any checked exceptions that they might raise in their API rather than catch them themselves. The application should abend on any exception, including IO exceptions on the compression work, database errors on load or persist, and any runtime exceptions. Service classes may catch exceptions in order to set a meaningful message then rethrow the exception out to the consuming class. The application must log the the exception message and a stack trace.

### Deliverables

1. A new Run class with main method, modeled on any of the classes in the edu.ucop.ers.generate.batch package.
2. A business logic class that does the work described above.
3. A report DTO if the existing ReportStructure class is not workable.
4. A payroll detail DTO if the existing DrillDownStructure is not workable.
5. Unit tests on all the units of work listed above.

## Purge program

This logic will delete all rows in the report archive table and the index tables, for the selected reporting period. It will be an offline batch program. The program must validate that the reporting period in question is currently in Archive state. At the end of a successful run, the program must update the reporting period’s status to Purged.

### Deliverables

1. Business logic class
2. Three new constants added to ERSConstants to reflect the new possible reporting period statuses
3. Driver class
4. Unit test on business logic

## Admin View

The application will add a new system admin view, protected by its own access point resource, where the user can enter search criteria, with an entry control for each of the index attributes. The application will return a list of matching reports. The user can then select the reports to download. The user must be able to also indicate which reports should include the payroll detail view.

Each searchable attribute of a report will have its own control on this view:

|  |  |
| --- | --- |
| Attribute | Control Type |
| Employee Name | Text input with typeahead functionality |
| Employee ID | Text input with typeahead functionality |
| Period Name | Drop down populated with all possible values |
| Account Org ID | Text input with typeahead functionality |
| Account Org Name | Text input with typeahead functionality |
| Fund Org ID | Text input with typeahead functionality |
| Fund Org Name | Text input with typeahead functionality |

The results table should replicate the existing paging user experience used elsewhere in the app, with switchable number of records per page, forward and back one page, and jump to start/end.

The search controls should be styled in a manner similar to the existing admin views.

The admin view will be supported by an action class, that handles:

* Initial presentation of the admin view
* Search
* List presentation
* Export

There will be two form objects, one for the search, and one for the list.

The link to the archive admin view should be conditionally displayed, only if there is at least one reporting period in the system in archived state.

### Deliverables

1. New jspf templates that contains search controls and section for display of results.
2. An action class to handle all four possible user requests, modeled on AddEmpIncludeListAction, including the admin access point check at the top of each action method. Action methods include:
   1. Display archive search view
   2. Execute search, returning list of matching items
   3. Export of selected reports

This class should go in a new package: edu.ucop.ers.admin.struts.action.reports.archive

1. An action form to carry search parameters and selected reports. This class should go in the existing edu.ucop.ers.admin.struts.form package.
2. Updates to struts-config.xml file to add new form and action mapping.
3. A javascript function to implement the search
4. Modification of the ERS\_INSTALL\_BASE scripts for DB2, Oracle, and SQL Server to include a new admin access point with:
   1. resource ID ‘ArchivedRpts’
   2. permission ID ‘AccessAchive’
   3. permission name ‘Access archived report search and display’
   4. role ID ‘AccessAchive’
   5. Role name ‘Archive searcher’
   6. Function ID ‘COORD’

Insert statements will include:

1. ERSPermission
2. ERSResource
3. ERSUseResource
4. ERSUseFunction
5. ERSRole
6. ERSRolePermission

The same set of statements must be included in the set of ERS\_R10-11\_UPGRADE\_\* scripts.

## Search Implementation

The action class should delegate to a search provider. The delegate API will include one method:

1. List<ArchiveReportDTO> search( ArchiveSearchForm searchForm )

The search should be written in HQL. The search should only join the index table when one of the multiple value attributes is part of the search.

### Deliverables

1. New search delegate class, placed in a new package, edu.ucop.ers.admin.utility.reports.archive
2. The HQL necessary to support querying by all possible search fields.
3. New data transfer object that encapsulates the results list display fields of found reports:
   1. Emp Name
   2. Emp ID
   3. Period ID
   4. Period Name
4. Unit test on the search operation

## Archived Report Display

When the user posts the form on the list of matching reports, the app will select all the the binary report DTO objects related to the requested aerCd, uncompress, order by version number, pass through the JasperFillManager.fillReport() method followed by the JasperExportManager.exportReportToPdf(jasperPrint) PDF exporter, and then push the PDF bytes back to the browser. Use edu.ucop.ers.admin.struts.action.reports.compliance.DisplayCompReportAction as the model for the latter operation. The strategy is to get the output stream associated with the response, write all the bytes from each report directly to it, then flush and close it to complete the request.

The application will require two new JRXML templates laying out the report data for display, one for the report, and one for associated payroll detail. The layout of these template should be modeled on the existing print versions of these items.

### Deliverables

1. Action class
2. JRXML templates for report detail and payroll detail
3. Update to struts-config.xml to map new action
4. Unit test on query operation

## Web Application UX Changes

The report list and individual report views must filter archived and purged reporting periods from their drop downs. The reporting report admin view must include the new archive status attribute in its display of reporting periods.

### Deliverables

1. Modification of report actions that populate the reporting period filter drop down on the report list and report views.
2. Modification of the admin reporting period views to include the archive status attribute.

# Revision History

|  |  |  |
| --- | --- | --- |
| Version | Date | Notes |
| 1.0 | 2013-10-16 | Initial |
| 1.1 | 2013-10-17 | Add primary keys to psuedo DDL; use existing period status field on reporting period, adding constants to reflect new possible values; note existence of builder classes for DTOs. |
| 1.2 | 2013-10-17 | Add struts mapping as deliverable to any task requiring new forms/actions; discussion of batching strategy in archive program task; JRXML templates added to search display task. |
| 1.3 | 2013-10-17 | Strike period name from index; conditional display of archive admin view on system admin view; employee ID varchar(10) to support PeopleSoft HCM EMPLID under UCPATH. |
| 2.0 | 2013-12-16 | Reflect new reqluirements to capture all reports and all comments. |