CIS Backfill Documentation

CIS Backfill Documentation

Please note that this page and related pages are being developed as part of the CIM Courses Project and are subject to change.

This article describes backfilling of data from the Container/Template Subject Structure to CIS (Curricular Information System). The CIS Backfill was implemented as part of the CIM Courses project.

Purpose

Legacy CIS application is replaced by CIM Courses application, but legacy CIS tables are still being used by Scheduling, Online Subject Listing (OSL) applications etc. The CIS Backfill is implemented so that downstream systems that rely on legacy CIS tables would not be impacted after its replacement by CIM Courses.

Background

Front-end UI applications that modify subject information like SCASUBJI, CIM Courses etc. calls Subject Management API to save the subject data in Container/Template structure (CTSS). While saving subject information in CTSS, API also creates entries in *subject_backfill_queue* table (Queue table) so that data can be backfilled to MITSIS and CIS tables.

- CIS tables store subjects in approved and proposed states.
- Standard Subjects (*subject_container. subject_type = "Standard"*) are saved in CIS tables. Administrative and Cross Registration subjects are NOT stored in CIS.
- Changes made in SCASUBJI application are not stored in CIS.
- Following are the important tables used by CIS application -
  scrci_proposal - stores main subject details like titles, attributes, grading modes and units of a subject.
  scrci_seminar - contains seminar title, content and faculty information.
  scrci_cluster - stores renumbered subject keys, equivalency, meets with/scheduling relationship and cross-list information of a subject.
  scrci_term_plan - contains offered terms and duration of a subject and also instructors assigned.
  scrci_bulletin - stores information which is required for Online Subject Listing application (OSL) and Course Catalog.
  scrci_url - contains url of the subject's web page.
  scrci_warehouse - contains information compiled from multiple CIS tables and many jobs and applications use this.

CIS Backfill

- CIS backfill imports data into CIS tables.
- Table mapping between container/template structure and "old" CIS is explained in Table Counterparts - Old Structure to New Structure.
- CIS Backfill receives data to import from a staging table *subject_backfill_queue* (Queue table) and its helper table *subject_backfill_usedkey* (UsedKey table). Queue table and UsedKey table are populated by Subject Management API and CIM Inbound Feed .
- Subject Management API saves **approved** subject information in container/template tables (CTSS) and also creates entries in Queue table and UsedKey table so that data can be ported to legacy CIS tables and "old" MITSIS tables. Please note that one API call could create multiple queue records in subject_backfill_queue.
- CIM Inbound Feed calls Subject Management API to save approved subjects, but creates Queue table records directly for proposed

subjects.

- CIS Backfill ports data using CIS Backfill Batch Process . An http endpoint called Testing CIS Backfill Endpoint also exists which can import data to CIS. This endpoint only processes one queueId at a time and is mainly used by automated tests and for debugging purposes.
- **CIS backfill ports data to CIS tables from queue records only and does not read CTSS tables directly.** More information about structure and working of Queue table can be found at Queue section in Backfill Documentation .
- UsedKey table works in tandem with Queue table to help with proper processing of MITSIS and CIS backfill. When queue table is populated, usedKey table is also populated with the main subjectKey and all the related subject keys used in the queue record - main subject, equivalencies, subject relationships and cross lists of the subject.
- CIS Backfill processes a queue record only if none of the related subjects in usedKey table for the queue record, had errors logged while processing. This is because when edits are not ported properly into CIS, any of its equivalents, cross lists etc. also should not be ported so that the integrity of the subject is maintained.
- **Subjects for current Academic Year will be backfilled to CIS for Proposal Year.** ie. if a subject is edited for current year say 2020SU and if Proposal year (the year for which subject proposals are created by CIM Courses - also called Catalog year) is 2021FA, this subject will be backfilled to Proposal Year 2021FA in CIS.
- Only Standard Subjects for which the Effective From Term (*subject_template.effective_from_term*) is in the current Academic Year or the Proposal Year will be backfilled to the CIS tables. This condition is implemented in CIS Batch Process using *subject_mgmt_config* value ranges between **cisBackfillAllowedFromYear** and **cisBackfillAllowedUptoYear**. Please note that if CIS is backfilled using 'Testing CIS Backfill endpoint', effective_from_term condition is skipped and subject will be backfilled irrespective of the year for which the edit was done.

Backfill Data Flow Diagram

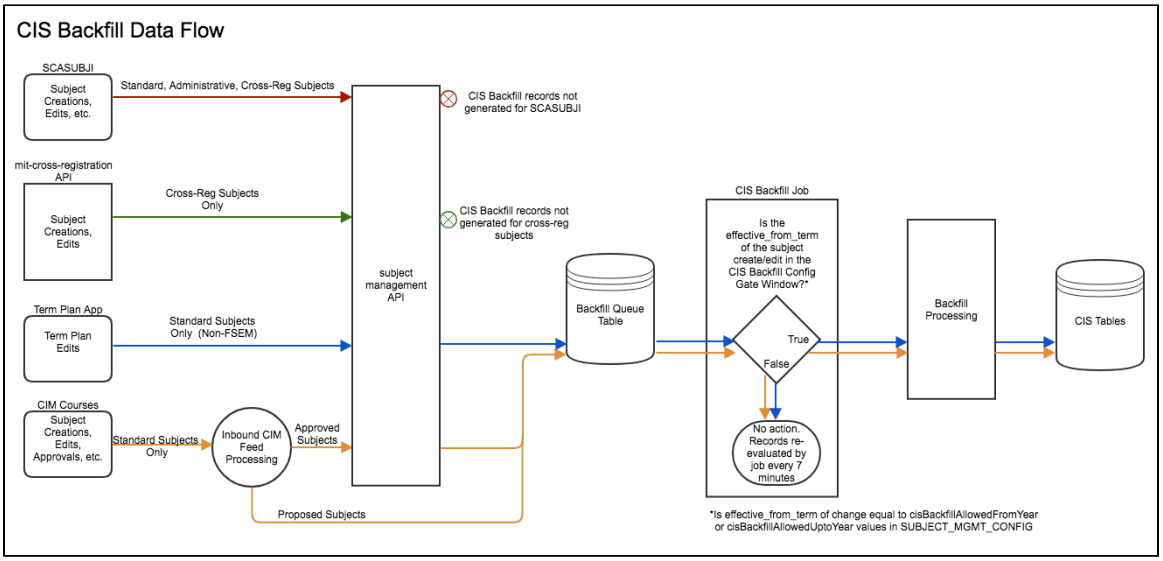


*Diagram up-to-date as of 2/28/2019

Technical Documentation

CIS Backfill is implemented using Mule flows in Anypoint Studio IDE. Both 'CIS Batch process' and 'Testing CIS Backfill endpoint' uses the same flows and hence processing logic is the same for both except for how it is called. Former is a batch process and picks up records from database at regular intervals (configuration detail here) while the endpoint is calls one queueId at a time. When CIS batch picks up records effective_from_term condition is also checked while endpoint ignores this condition and tries to backfill any queueId given to it as long as it is a Standard subject.

When a subject is edited/created, multiple queue records may be created in *subject_backfill_queue* table. All the generated record(s) need to be processed to reflect the status of the subject correctly in CIS or MITSIS. If any queue record has a processing error in CIS all subsequent records for the subject and its related subjects will be blocked and will not be processed until the error is cleared and subject_backfill_error table is cleared for the subject.

CIS Backfill uses legacy CIS oracle stored procedures (with minimal changes) so as to backfill according to the legacy CIS business rules. There are some new stored procedures too as explained under Stored Procedures section below.

In CIS, a subject is said to be in proposed state if the highest version of the subject key has *version_status* = 'SU' in CIS tables. Also a subject is said to be in approved state if *version_status* = 'OK'

A major change in legacy business rule is that a subject number need not be archived before it can be re-used.

Implementation

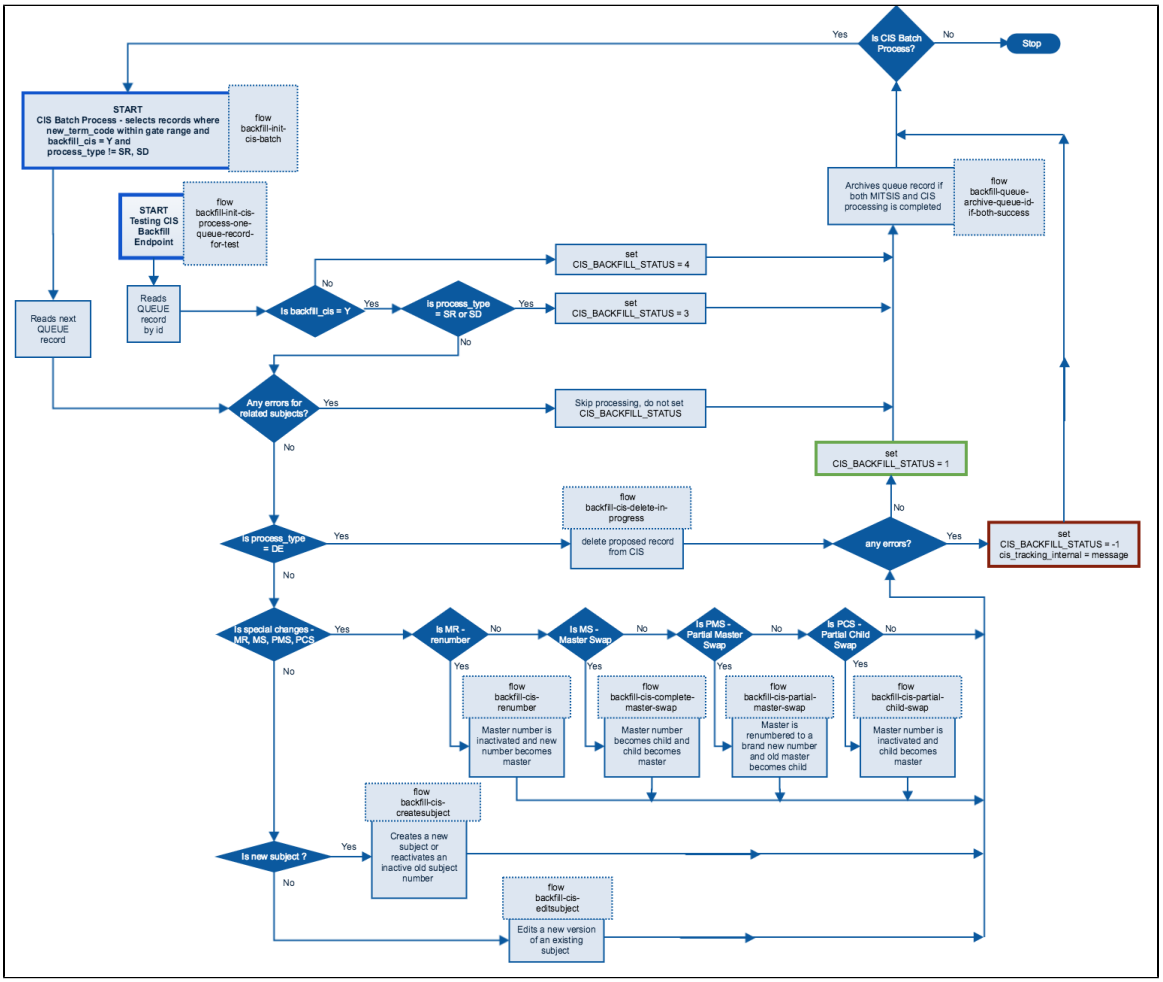
In the mit-subjects application, the XML files that contains the flows used in CIS backfill processing are:

backfill-initiators.xml
backfill-cis.xml
backfill-main-processor.xml
backfill-queue.xml
backfill-error.xml

The top-level flow for :

CIS backfill batch is backfill-init-cis-batch flow
CIS Testing endpoint is backfill-init-cis-process-one-queue-record-for-test flow

Backfill Processing Logic



* Updated as of Feb 2019. Source for diagram(s): https://wikis.mit.edu/confluence/display/EduSys/CIS+Backfill+Processing+Logic

- CIS backfill processing processes one *subject_backfill_queue* record (queue record) at a time, in ascending order of *subject_backfill_queue_id*.
- If *cis_backfill_status* = null or 2 means that backfill was not done on the record and is ready to be processed. cis_backfill_status =1 indicates already processed, -1 error etc.. More details under Queue section in Backfill Documentation.
- If *backfill_cis* = *Y* and backfill_process_type != (SR,SD) CIS Backfill is attempted on the queue record. Additionally *CIS batch process* checks if *new_term_code* column values in queue recored is within the range of gate config values before attempting CIS backfill.
- SCASUBJI changes are marked with *backfill_CIS* = *N* by Subject Management API and are not backfilled to CIS. This is because legacy CIS only saved Proposal year changes in CIS tables and so the stored procedures only support that functionality. SCASUBJI can edit subject for any year and there were issues in versioning logic in making current year and prior year changes using CIS stored procedures. So the team decided to be consistent and not backfill anything from SCASUBJI - since Proposal year changes can be made via CIM Courses (and the Admin Save function) if necessary.
- all subjects are backfilled to CIS only for Proposal year. ie. term of the subject edited for current academic year will be overridden by CIS Backfill to Proposal year (one year ahead). This subject can then be back dated (*backfill_process_type=BD*) to the current academic year

if needed.

- In Queue record, modified data is saved in JSON format in new_data column and subject details before modification is saved in previous_data column. Note that these columns have different JSON structure according to backfill_process_type.
- old data and new data is read and compared to determine if the queue record was a special change operation - renumber (MR), master swap (MS), partial master swap (PMS), partial child swap (PCS) (see more details here) or whether the operation is a mere new subject creation or editing an existing subject or deletion of a proposed subject.
- CIS system rule allows only one proposed change per subject (in scrci_proposal table) and hence if a subject is changed more than once in proposed state, only the last state is saved in CIS tables. Deleting a proposed change is implemented in flow 'backfill-cis-delete-in-progress' by_ _calling legacy SCRCI_POST_PRC procedure with in_request parameter as 'Delete'.
- if *backfill_process_type = DE* , the proposed subject record is deleted. Approved subjects can only be inactivated and cannot be deleted in CIS.
- if operation = MR (renumber), mule flow 'backfill_cis_renumber' calls stored procedure SP_CR_CIS_SUBJECT_BACKFILL on the old subject number with in_request_type parameter as 'NewVersion.IW.Y' and virtual column optional parameter NEWSUBJKEY= new subject number.
- if operation = MS (master swap), mule flow 'backfill_cis_complete_masterswap' calls stored procedure SP_CIS_BACKFILL_MASTER_SWAP.
- if operation = PMS (partial master swap), mule flow 'backfill_cis_partial_masterswap' adds the old number as cross listed child and then does a master swap operation.
- if operation = PCS (partial child swap), mule flow 'backfill-cis-partial-child-swap' does master swap and then remove the old master which is a child after swap operation.
- if old data is null and new data exists, the subject is probably a brand new subject. So verify if the new subject number is valid using legacy stored procedure SCRCI_NEW_RECORD_PRC. If the subject number is valid, create the subject using SP_CR_CIS_SUBJECT_BACKFILL with in_request_type parameter as 'Add' .
- if editing an existing subject is the operation, then call SP_CR_CIS_SUBJECT_BACKFILL with in_request_type parameter as 'NewVersion.IW.Y'.
- master calculation - during all these operations if the subject has an equivalent (EQ) or scheduling relationship (MW), then *scrci_cluster* should have an entry. In *scrci_cluster* one of the EQ/MW has to be the master. The master calculation is done so as to mimic legacy CIS application as much as possible. In legacy CIS application, master was always known to the user and hence EQ and SR child subjects in the cluster was always added to the master only. To simulate this in the Mule flow, *scrci_cluster* entries are checked to see if a master already exists. If a master exists, then that master is edited using stored procedure SP_CR_CIS_SUBJECT_BACKFILL to add/bookend equivalents and scheduling relationships. If a master does not exist and if this is a new cluster, then the current subject being edited is made the master and EQ/SR is added to this subject.
- *scrci_proposal.rationale* will be standard text 'Backfilled from CIMCourses' for all subjects. The actual rationale for any change is saved only in CIM.
- *scrci_proposal.compare_version* will not be populated. This field was used for workflow management in legacy CIS application.
- attributes are mapped according to CIS business rules as explained in Attribute handling section.
- during CIS processing details are also written to *subject_backfill_log* table.
- queue record which is processed successfully (not skipped) by both CIS and MITSIS is archived to *subject_backfill_queue_archive* table. Stored procedure SP_MOVE_TO_SUB_BACKFILL_Q_ARCH implements the archival process and also stores comma separated subject keys from usedKey table for historical purposes.

Stored Procedures

- Legacy stored procedures were reused as much as possible to backfill CIS in-order to preserve existing logic. These stored procedures and CIS tables are prefixed with *"scrci_"* . Information about legacy procedures can be found in attached document - https://kb.mit.edu/confluence/download/attachments/157354806/CIS_stored_procedures.pdf;.

- Some of the important procedures are explained below -
    - SCRCI_POST_PRC - is the head of legacy stored procedure tree which initiates a subject/seminar creation or editing. This calls other procedures as needed according to "in_request" parameter value - 'Add', 'Delete', NewVersion'. To make this procedure versatile "virtual_columns" parameter is used which is actually a long list of optional parameters.
    - SCRCI_PROPOSAL_PRC - called by SCRCI_POST_PRC for subject creation and modifications.
    - SCRCI_SEMINAR_PRC - called by SCRCI_POST_PRC for seminar creation and modifications.
    - SCRCI_TERM_PLAN_PRC - called by SCRCI_POST_PRC for term plan only changes.
    - SCRCI_STATUS_CHANGE_PRC - simulates workflow state changes in CIS. IW -> PR -> DR -> SU -> OK
    - SCRCI_NEW_RECORD_PRC - called before a new subject is created to check if the subject number is valid for the term.
    - SCRCI_PROPOSAL_DELETE - deletes a proposed subject.
    - SCRCI_VALIDATE_CLUSTER_CHILD - procedure which checks if the child is valid in an equivalent or meetsWith cluster.

- New stored procedures were also written for CimCourses project. Some were written as a wrapper to simulate stepping through different workflow stages in CIS or to streamline operations like master swap which was done manually earlier.
    - SP_CR_CIS_SUBJECT_BACKFILL - simulates progressing through old CIS workflow calling multiple legacy stored procedures. This is the main procedure called for creating or editing a subject/seminar. "virtual_columns" parameter in this is the exact same expected by SCRCI_POST_PRC and these optional parameters are built in 'getVirtualColumnsForSP' method call in CISSubject java class.
    - SP_CIS_BACKFILL_MASTER_SWAP - implements a master swap. Since old CIS application did not have a method to do this easily, this was a manual process and is now replaced with this stored procedure.
    - SP_BACKFILL_SCRCI_URLS - scrci_urls were manually updated from a separate application and uploading to tables was a manual process. A new stored procedure was added so that URLs can be updated from SCASUBJI.
    - SP_BACKFILL_MITSIS_MW_CLUSTER - this is a helper procedure which is used to reset cluster_type in *scrcu_cluster* table for meets-with subjects which does not have a child.
    - SP_CR_SUBJECT_BACKFILL_ERROR - procedure which inserts errored subject keys into *subject_backfill_errorkey* table so that once a queue record has an error, all other queue records for those subjects are kept on hold.
    - SP_MOVE_TO_SUB_BACKFILL_Q_ARCH - queue records whose CIS and MITSIS backfill processes are completed are

moved to *subject_backfill_queue_archive* table for record keeping log. This moving logic is done by this procedure.

Error Handling

- When CIS Backfill Processing encounters an error, cis_*backfill_status* is set to -1 and detailed error is written to *subject_backfill_queue.cis_internal_tracking* column. All subject keys for the queueId in *subject_backfill_queue_usedkey* table is copied to *subject_backfill_errorkey* table so that subsequent queue records which refers to any of those subject keys are blocked from processing.

- User friendly error messages are logged into *subject_backfill_log* table for future use.

- If an unexpected error occurs during the processing of the data feed, email is sent to a list (cim-courses-support@mit.edu as of Feb 2019) with the subject line “CIS Backfill Error (prod environment)”. The body of the email message contains details about the error.
  The property that defines the email recipient address is **backfill.email.to**
  The property which defines if email is to be sent immediately when an error occurs is controlled by **backfill.error.email.send.instantly (true/false)**

- A digest email is sent to list (cim-courses-support@mit.edu as of Feb 2019) with summary of all CIS errors once a day. This email has a subject line “CIS Backfill Errors (prod environment)”. The body of the email lists subject keys and error encountered for each subject. This is implemented as batch process in mit-subjects application hosted in Cloudhub as **backfill-init-batch-send-error-email Poll**
  The property that defines the email recipient address is **backfill.email.to**
  The property that defines the schedule and time the digest email is set is controlled by **backfill.digest.errors.email.schedule** in cron like format
  (eg: 0 0 10 ? * MON-FRI)

Attribute Handling

The handling of subject attributes is documented on the main Backfill Documentation page

Downstream Systems

- Scheduling/UniTime - data from CIS (both proposed and approved) is pulled into a staging table, SCHED.SCHED_SUBJECT. That data is disseminated to other tables in the SCHED schema which are then used by UniTime.
- Online Subject Listing - takes all of its data from CIS tables except for faculty data and URL. Faculty data and URL are pulled directly from the CTSS.
- Data Warehouse - data from CIS is exported into a table called SCRCI_WAREHOUSE. Data from SCRCI_WAREHOUSE is then imported into the Data Warehouse. It is assumed that downstream systems pull data from both SCRCI_WAREHOUSE and the Data Warehouse.

CIS Testing Environment with User Interface

As of 4/3/2019, a CIS user interface with editing privilege was still being maintained for testing in the sched-dev environment. Link: https://student-sched-dev.mit.edu/cgi-bin/stv_custom_menu.sh?Application=CISP&Template=CISP_main_menu

FAQ

Q: **A change made in CIM Courses is not reflected in CIS. What gives?**
A: There are several reason why a change made in CIM may not update the CIS tables. Here are some things to check:

- Changes in CIM are only processed once per day, early in the morning. Ensure that the change was made yesterday.
- The CIM Courses Inbound Feed has many validations that may hold up the processing of a subject. Validation errors are sent to cim-courses-support@mit.edu as of Feb 2019.
- The record created to backfill the subject data to the CIS tables encountered an error. These errors are emailed to cim-courses-support@mit.edu in an email with a subject of "CIS Backfill Errors (prod environment)"

Q: **A change made in SCASUBJI is not reflected in CIS. What gives?**
A: Changes made in SCASUBJI are not backfilled to CIS.

Subject Management Documentation Index

The Subject Management Documentation Index is the central listing for documentation pertaining to Subject Management.