MITSIS Backfill Documentation

MITSIS 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.

- MITSIS Backfill Documentation
  - Purpose
  - Background
  - MITSIS Backfill Data Flow Diagram
  - Backfill Details
    - 5-Year MITSIS Equivalency Rules
    - **Technical Documentation**
      - Implementation
      - Backfill Processing Logic
      - SR Backfill
        - Master rule - Assignment of Master for subjects in Scheduling Relationship cluster
      - SR Backfill Processing Logic
  - Error Logging
  - SCRSU_VAR Cross-Registration Master Mapping
  - Data Validations
  - Attribute Handling
  - FAQ
  - Subject Management Documentation Index

This article describes the backfill of data from the Container/Template Subject Structure to MITSIS. The MITSIS Backfill was implemented as part of the CIM Courses project.

Purpose

The MITSIS Backfill was implemented because the scope of the CIM Courses Project did not include repointing downstream systems from MITSIS to the new Container/Template Subject Structure (CTSS). Since the Container/Template Subject Structure would become the source for all subject data, there was a need to backfill that source data to "old"MITSIS tables. Here after in this document MITSIS tables refer to "old" MITSIS tables.

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 creates record(s) in *subject_backfill_queue* table (Queue table) also so that data can be backfilled to MITSIS and CIS tables.

- Standard, Administrative and Cross Registration subjects are stored in "old" MITSIS tables.
- MITSIS tables store subjects in approved state only. ie. subjects in proposed state are not stored in MITSIS.
- Following are the important tables used by MITSIS application -
  scbsu_key - indicates the terms for which a subject is valid.
  scrsu_var - stores main subject details like titles, units, meets with subject information and the department and school the subject belongs to.
  scrrtrm - indicates the terms for which a subject is offerred.
  screqiv - stores subjects which are equivalent to a subject. This also includes cross-listed subjects. More details on equivalency criteria in section - 5-Year MITSIS Equivalency Rules.
  scrgmod - grading modes of a subject.
  scrattr - stores attributes of a subject.

MITSIS Backfill Data Flow Diagram

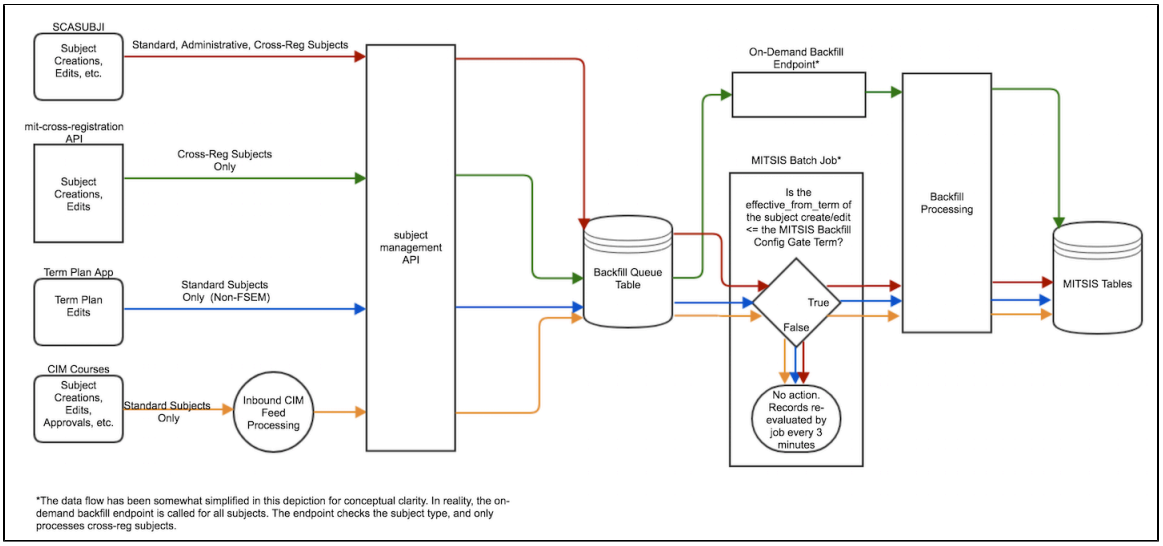


*Diagram up-to-date as of 2/28/2019

Backfill Details

- MITSIS Backfill imports subject data in CTSS to MITSIS tables indirectly using subject_backfill_queue table .
- MITSIS Backfill ports data using Batch Process most of the time. Cross Registration subjects need to be backfilled at real time and hence Subject Management API directly calls On-demand endpoint for this subject type alone . An http endpoint Testing Endpoint also exists which can import data to MITSIS. This endpoint only processes one queueId at a time and would need to be called multiple times to simulate an API call and is mainly used by automated tests and for debugging purposes.
- Backfill Process 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. Please note that **one API call can create multiple queue records in *subject_backfill_queue***. CIM Inbound Feed calls Subject Management API to save approved subjects and creates Queue records directly for proposed subjects.
- **MITSIS backfill ports data to MITSIS 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. UsedKey table is used to determine subjects which are related to the main subject so that if an error happens to any one of these subjects, backfill can be suspended for all the related subjects and thus integrity of data is maintained in MITSIS. UsedKey table is 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. When an error happens during backfill processing, subjectKeys for queueId (includes related subjects) are inserted to *subject_backfill_errorkey* table to stop processing those subject keys.
- MITSIS Backfill processes a queue record only if none of the related subjects in usedKey table for the queue record has errors logged in *subject_backfill_errorkey* while processing.
- Table mapping between container/template structure and MITSIS is explained in Table Counterparts - Old Structure to New Structure.
- Background information on MITSIS tables - Records are inserted into MITSIS tables only when some column information is different. ie. When an attribute is added for a subject, only *scrattr* table will have the added record with attribute effective the term when the subject was edited. The main subject table *scrsu_var* will not have a record for the change as attribute change does not affect any columns in scrsu_var. In short, when change happens, only the specific table where the change occurred has an additional record for edited term in MITSIS tables.
- **Only subject changes for current academic year or earlier years are backfilled to MITSIS**. This is enforced using gate config values. This is implemented in MITSIS Batch Process using *subject_mgmt_config* value <= **mitsisBackfillAllowedUptoYear**. This means that changes for Proposed year is backfilled to MITSIS only when that year rolls in. ie. if current term is 2020FA, changes to subject meant for Proposal Year (Catalog year) 2021 will be backfilled only in 2021 and not now in 2020FA. Please note that if MITSIS is backfilled using On-demand endpoint and Testing Endpoint , this check is skipped and subject will be backfilled irrespective of the year for which the edit was done.
- screqiv table stores equivalencies between subjects in MITSIS and equivalencies are valid for 5 years after the equivalency ends. These rules are explained in the section below. Cross-listed subjects are considered equivalent in MITSIS.

5-Year MITSIS Equivalency Rules

Applicable System Rules:

1. If Subject A is EQ with Subject B - and Subject B is deactivated, the two numbers will stay equivalent to each other for 5 years in MITSIS.
   This rule also applies to the removal of cross-lists and SWEs.
   a. Example: Subject A and Subject B had an EQ that began in 2010FA. Subject B was then deactivated - last active term was 2013SP.
      So the EQ persists in MITSIS upto 2018SP. The equivalency should not be in place for 2018SU.
2. Subject A is EQ to Subject B - and Subject B is deactivated. In a subsequent proposal year, Subject C is added as an EQ to Subject A. In this case, Subject C will also get Subject B as an EQ in MITSIS until Subject B has been inactive for 5 years. This rule also applies to the removal of cross-lists and SWEs.
   a. Example: Subject A and Subject B had an EQ that began in 2010FA. Subject B was then deactivated - last active term was 2013SU.
      Subject C was then added as an EQ to Subject A in 2015FA. Subject C would then be equivalent to Subject B for 2015, 2016, 2017, 2018. Subject B will not be in place from 2019FA.
3. If a Subject is renumbered, the previous number and the current number will stay equivalent to each other for 5 years in MITSIS.
   a. Example: Subject 1.88 is renumbered to 1.702, effective 2014JA. The EQ persists in MITSIS upto (including) 2018JA. The equivalency should not be in place for 2018SP.
4. If Subject A is cross-listed to Subject B and if cross-list to Subject B is terminated, the two numbers will stay equivalent to each other for 5 years in MITSIS since cross-listed subjects are considered equivalent in MITSIS.
   a. Example: Subject 2.34 is cross-listed to 3.78. If cross-listing to 3.78 is removed effective 2014SP, EQ persists in *screqiv* table upto 2019JA. Equivalency will not be in place for 2019SP.

Please note that the 5-year calculation is addition of 5 years taking term into consideration. For example, if a subject was renumbered effective 2015SU, the previous number will be included in *scrquiv* sets for changes with an effective_from_term < 2020SU (not 2020FA).

Please refer to the Subject Equivalency Removal Matrix for more detail about how actions on EQs are handled in each system.

Technical Documentation

MITSIS Backfill is implemented using Mule flows in Anypoint Studio IDE. MITSIS Batch process, On-demand endpoint and Testing endpoint uses the same flows and hence processing logic is the same for all except for how it is called. Former is a batch process and picks up records from database at regular intervals while the endpoints execute one queueId at a time. When MITSIS batch picks up records, effective_from_term condition is also checked while others ignore this check and tries to backfill any queueId given to it as long as it is an approved 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 MITSIS. If any queue record encounters a processing error, all subsequent records for the subject and its related subjects will be blocked and will not be processed until the error is fixed manually and *subject_backfill_error* table entries for errored queueId are cleared.

Implementation

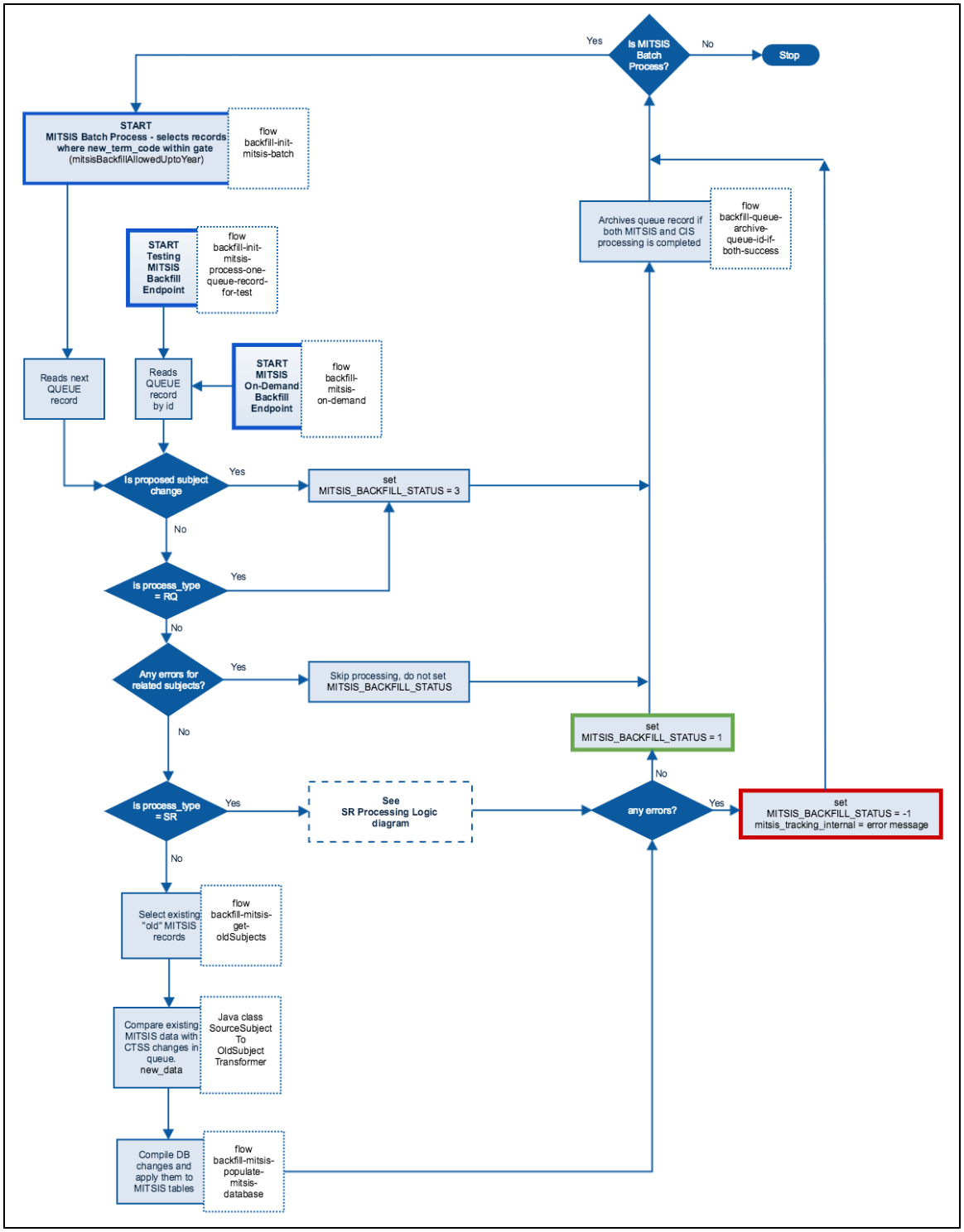
In the mit-subjects application, the XML files that contains the flows used in MITSIS backfill processing are:

```
backfill-initiators.xml
backfill-main-processor.xml
backfill-mitsis.xml
backfill-mitsis-db.xml
backfill-queue.xml
backfill-error.xml
```

The top-level flow for :

MITSIS backfill batch is *backfill-init-mitsis-batch* flow
Testing MITSIS Backfill endpoint is *backfill-init-mitsis-process-one-queue-record-for-test* flow
MITSIS Backfill On-demand endpoint is *backfill-mitsis-on-demand* flow

Backfill Processing Logic

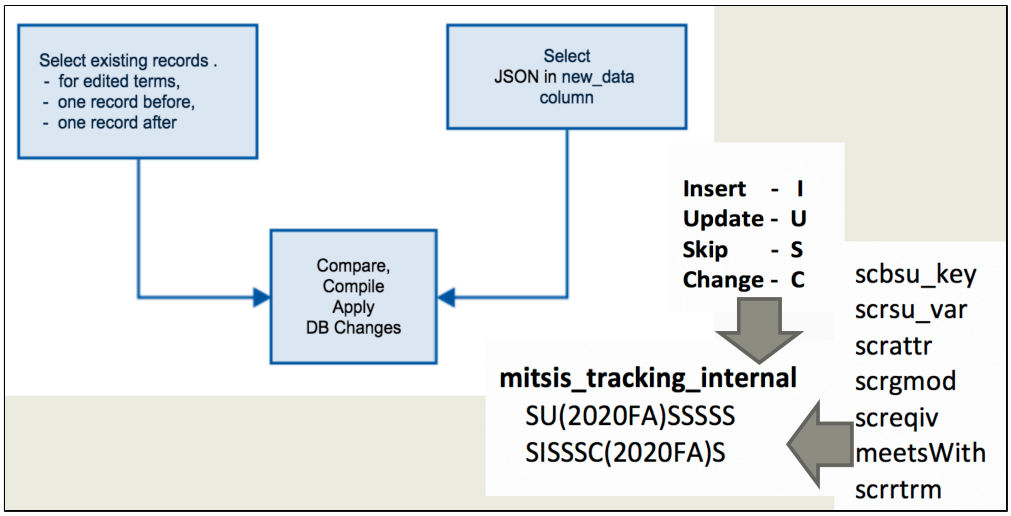


*updated Feb 2019

- MITSIS backfill processing processes one *subject_backfill_queue* record (Queue record) at a time, in ascending order of *subject_backfill_queue_id*.
- In Queue record, state of the subject before the subject is edited is stored in *previous_data* column and state of the subject after the change is available in *new_data* column. Both are in JSON format. Note that these columns have different JSON structure according to *backfill_process_type* value.
  Show/hide sample JSON of new_data column with *backfill_process_type* = null (regular) queue record
- If *mitsis_backfill_status* = null or 2 indicates that backfill was not done on the record and the record is ready to be processed. More details under Queue section in Backfill Documentation. *subject_backfill_log* table will have details of record processing.
- If *proposed_change_flag* = Y or *backfill_process_type* = ('RQ' , 'DE') , MITSIS backfill is skipped since proposed subjects are not backfilled to MITSIS or because RQ, DE process types are used only for CIS backfill. For these *mitsis_backfill_status* is set to 3 indicating

that MITSIS backfill is ignored. For all other process types, MITSIS backfill is done. SR type record is used to backfill meets with/scheduling relationship subjects into *scrsu_var* table and has different logic as explained in SR Backfill Processing Logic section.

- Comparison and MITSIS backfill processing logic of a queue record with _backfill_process_type <> SR (flow -backfill-main-mitsis-process-regular) is as follows:
  - Step 1 - Backfill reads the data in *new_data* column and transposes those to MITSIS rules and saves it in flowVars.sourceSubject
    - Subjects with Arranged units is transposed with "1" in the Lab Units field and all other units as
    - Lab Institute Requirements are backfilled as explained in LAB/LAB2/LB Attribute Mapping in Backfill Documentation.
    - Attributes are mapped according to MITSIS business rules as explained in Attribute handling section.
    - Rest of the fields do not have any special transposing rules and is ported as is.
  - Step 2 - Backfill processing collects existing data for the subject already in MITSIS tables - general information of the subject l(titles, units etc.), attributes, grading modes, equivalencies, subject relationships and terms offered. These data is gathered for
    - for the edited terms (between effective_from_term and effective_thru_term of the template indicated in *new_data*).
    - one subject record for the term before the_ effective_from_term_ if exists
    - one record after the *effective_thru_term* if it exits
  - Step 3 -Comparison - subject information in new_data column in Step1 is compared with Step2 data already in MITSIS tables. Comparisons are done by java class **SourceSubjectToOldSubjectTransformer** and helper classes. The transformer returns a compiled list of changes in **List<BackfillSubject>** with **BackfillAction** type set which indicates whether insert or update or delete of the data should happen for each of the MITSIS tables. These actions are also saved in *mitsis_tracking_internal* column for debugging later.
    - For example: SU(2020FA)SSSSS indicates scbsu_key table is skipped (no change), in scrsu_var 2020FA record is updated with new data, scrattr is skipped, scrgmod is skipped, screqiv is skipped and meetsWith information in scrsu_var (cluster_type, subj_code_master, subj_numb_master) is skipped, scrrtrm is also skipped
    - 4 operations are - Insert (I), Update (U) with term specified, Skip (S), Change (C) with multiple records for the term specified (all records for the term are deleted and new set inserted)
    - Tables specified in order are - scbsu_key, scrsu_var, scrattr, scrgmod, screqiv, meetsWith (indicates SR/meetsWith related columns in scrsu_var - cluster_type, subj_code_master, subj_numb_master) , scrrtrm
  - Step 4 - Apply DB Changes - BackfillAction specified in Step4 is applied to database. Flows in backfill-mitsis-db.xml executes the action specified in BackfillAction for each of the BackfillSubject objects and the change is reflected in the MITSIS tables.
  - Step 5 - Logging - User friendly summary is also logged into *subject_backfill_log* table to indicate the actions performed in MITSIS tables for the subject in addition to subject_backfill_queue.*mitsis_tracking_internal*.



- *mitsis_backfill_status* column in set to 1 if processing is successful, but if an error occurs *mitsis_backfill_status* is set to -1 with error details set in *mitsis_tracking_internal* column. When an error occurs, subject keys from UsedKey table for queueId will be inserted into *subject_backfill_errorkey* table also to stop processing any related subject keys.

- Limitation of backfilling equivalents - screqiv is only backfilled for the edited term by MITSIS backfill.
  subject_container_equiv table in CTSS stores equivalents between containers for a range of time. While editing a subject from SCASUBJI UI, equivalency from term can be changed to start from a term earlier than the term when the subject is edited. To backfill equivalencies correctly in screqiv and retain those for 5 years, the history of all the subjects which were/are equivalent to the edited subject (including their cross-lists) are required since the subjects could have been renumbered, subject numbers swapped etc. previously. Due to complexity of processing and since this is an edge case, a backfill restriction was implemented so as to backfill only newly added equivalencies and remove existing equivalencies too, but only from the term when the subject is edited. ie. if a subject is edited in 2019FA, any new/remove equivalent subject changes are only backfilled from 2019FA. If this subject is edited to start an equivalency from 2018FA instead, in screqiv equivalency will be backfilled to start from 2019FA only. There is no restriction on already existing equivalencies of the subject though.

- Meets with /scheduling relationship information is backfilled to MITSIS for all terms even if a new SR starts from earlier term. More details in the section below SR Backfill

SR Backfill

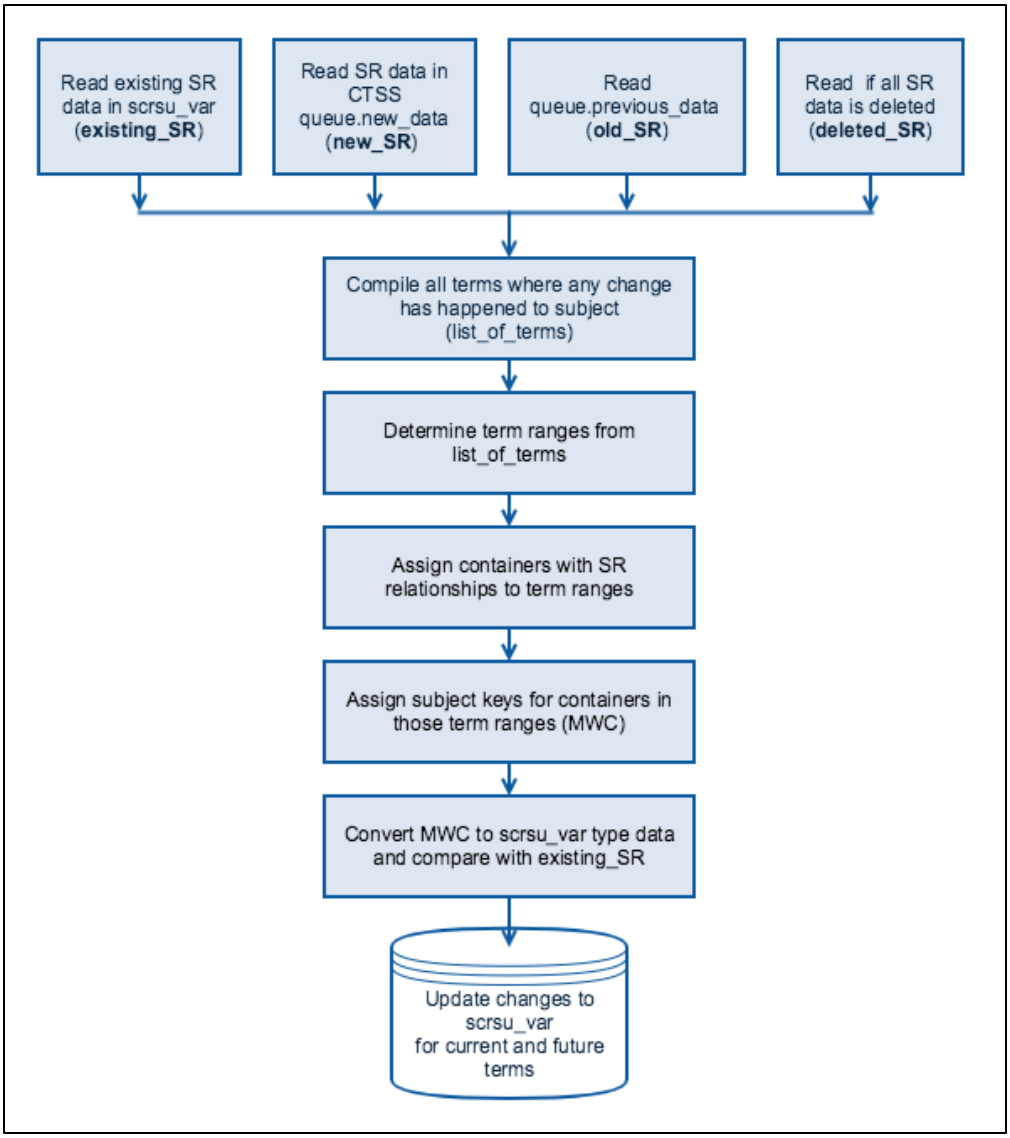
When subjects with scheduling relationships are created or edited, Subject Management API creates an SR (Scheduling Relationship) record in *subject_backfill_queue* table. This contains the history of scheduling relationships in JSON format in the *new_data* column and this record is processed differently. Please note that this record is created in addition to the regular queue record when the subject has SR relationships with other subjects.

- SR record is used to backfill meets with/scheduling relationships of subjects into scrsu_var table in MITSIS database. Please note that any other "old" MITSIS table is modified by SR processing.
- If there are any errors in *subject_backfill_errorkey* table for the subject being processed or its related subjects in *subject_backfill_queue_usedkey* table, then further processing is skipped.
- Data in in *previous_data* column and *new_data* column in *subject_backfill_queue* table is read and compared to check if there are any differences in scheduling relationships of the subject. *previous_data* column holds the history of scheduling relationships before the change and *new_data* column holds scheduling relationship information after the change.
- If no difference is found between *previous_data* and *new_data* scheduling relationships, then *mitsis_tracking_internal* column is set as *';* **No new scheduling relationships to be added***'* and further processing is skipped.
- If there are differences in scheduling relationships between *previous_data* and *new_data*, then the whole SR history in the columns are scanned and compared with existing data for the subject in *scrsu_var* table.Mule flow *backfill-main-mitsis-process-sr* implements SR Backfill Processing logic
- In MITSIS table *scrsu_var*, each scheduling relationship cluster has a master. This master has to be preserved as much as possible when a subject is edited unless graduate-undergraduate rule has been violated. This Master assignment rule is explained in section below. When a subject whose master assignment rule has been violated is edited, master can be modified from that term onwards to conform to the rule, but not for past terms.

Master rule - Assignment of Master for subjects in Scheduling Relationship cluster

- For edits to existing scheduling relationship(s) (SRs), if the previous master is undergraduate, it will be retained.
    - If the previous master is graduate and there are no undergraduate subjects in the cluster, then the previous master will be retained
    - If the previous master is graduate and there are one or more undergraduate subjects already in or added to the cluster, an undergraduate subject will become the master for previous and new data (lowest alphanumeric undergraduate subjects will be assigned as master)
- For new SRs between undergraduate and graduate subjects, the undergraduate will be assigned as the master
- For new SRs between two or more undergraduate subjects, the subject with the lowest alphanumeric subject number will be assigned as the master

SR Backfill Processing Logic



*updated Feb 2019

Data in SR related columns is read from *scrsu_var* table and is compared with transposed SR information passed in from subject_backfill_queue.new_data column. If there is any difference, scrsu_var record is modified from edited term onwards. So no change is made to past data in scrsu_var even if the subject does not follow the 'Master rule'.

- Step 1.0 -
  - **existing_SR** (old MITSIS type data) - Read existing SR data from scrsu_var table to create a baseline for comparison. Relevant columns read are *subj_code, subj_numb, effective_term, cluster_type, subj_code_master and subj_numb_master* in *scrsu_var* which determines scheduling relationships and the master subject in MITSIS.
  - **new_SR** (new_data type) - Read *new_data* column (JSON format) from *subject_backfill_queue* which contains the latest SR information which needs to be backfilled
  - **old_SR** (new_data type) - Read *previous_data* column from *subject_backfill_queue* which reflects the data before the change
  - **deleted_SR** (new_data type) - When all SR records are deleted, this element will have all the deleted values. This was done because new_data cannot be null and previous_data was added at a later time.

- Step 1.1 - list_of_terms - Compile all the terms (only terms are tracked in this step) where any change occurred for any of the subjects in SR cluster
  - Collect all the from and thru terms in new_SR, old_SR and deleted_SR data and add to list_of_terms. Do not worry about adding same term again and again since these will be weeded out while sorting.
  - Add in terms in existing_SR where any change in cluster_type or master or subjectLevel occurred in the cluster.
  - Sort the terms in ascending order and remove duplicates. These would be the terms where any SR cluster change (or MWC - meets with cluster change) would have happened.

- Step 1.2 - Determine term ranges from list_of_terms from the last step and assign subjects with scheduling relationship or meets with cluster (MWC) in the term range. Existing master in *scrsu_var* is also identified and noted for each MWC according to Master Assignment Rule . Since data is ported into container/template tables only from 2013FA, MWC groups for terms earlier than 2013FA are ignored and term ranges are conditioned for effective comparison.
  - The terms are analysed to create 'term ranges' to calculate the most granular term range when change occurred for any of the

subjects in the cluster.
    - Add containers from new_SR, old_SR and deleted_SR for each term range. There should not be an overlap of terms here because list_of_terms were compiled from these data too
    - Translate those containers to the subject keys pertinent to the range of terms from JSON information. Here also there should not be an overlap since terms where changes happened were also compiled into list_of_terms.

- Step 1.3 - Transpose MWC from last step to *scrsu_var* type data and compare with existing_data. Any changes before the term being edited are ignored (since we do not want to change any past data which does not follow master assignment rule) and only update the subjects from this term onwards to comply to master assignment rule.

- subject_backfill_log record created for queue_id with success message. Error handling is explained below.

Error Logging

When MITSIS Backfill Processing encounters an error (any queue record irrespective of backfill_process_type):

- *mitsis_backfillc_status* is set to -1 and detailed error is written to *subject_backfill_queue.mitsis_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 “MITSIS 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 MITSIS errors once a day. This email has a subject line “MITSIS 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)

SCRSU_VAR Cross-Registration Master Mapping

Institution	Description	Subj_Code*	Master_Subj_Code	Subj_Num	Master_Subj_Num
Harvard	Subject Code and Master Subject Code should be the same value. Master Subject Number should be “0000”.	HAB	HAB	1234	0000
Wellesley	Subject Code and Master Subject Code should be the same value. Subject Number and Master Subject Number should be the same value.	WED	WED	325	325
Mass College of Art	Master Subject Code should be “ MC”. Master Subject Number should be “0000”	MCA	MC	207	0000
Brandeis	Subject Code and Master Subject Code should be the same value. Subject Number and Master Subject Number should be the same value.	BR0	BR0	S12	S12

*Please note these are only example Subject Codes. Subject Codes for these institutions can vary.

Data Validations

MITSIS Data Validations are documented on the Subject Management Business Rule KB Page page

Attribute Handling

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

FAQ

Q: I was testing the MITSIS backfill and my test subject is not being processed (it remains in subject_backfill_queue). Why won't it backfill?
A: The effective term of the change may be greater than the SUBJECT_MGMT_CONFIG.mitsisBackfillAllowedUptoYear value. For example, if your test subject has an effective term of 2019FA but MITSIS may only be expecting subject changes for AY2018.

Q: I was testing the MITSIS backfill and my test subject edit processed but I'm not seeing a new record in SCRSU_VAR. What gives?
A: Unless your test contains a change specific to one of the SCRSU_VAR fields, the backfill will not create a new SCRSU_VAR record. For example, if your test is only adding the UROP attribute to a subject, no SCRSU_VAR fields would have changed - and therefore there is no need to create a new record in that table. Only SCRATTR will get the new record(s).

Subject Management Documentation Index

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