



# **SNOMED International Frequent Delivery public proposal**



Document Status	<b>EXTERNAL REVIEW</b>
Document Version	0.25

© 2021 International Health Terminology Standards Development Organisation. All rights reserved. SNOMED CT® was originally created by the College of American Pathologists.

This document forms part of SNOMED CT® distributed by International Health Terminology Standards Development Organisation, trading as SNOMED International, and is subject to the SNOMED CT® Affiliate License, details of which may be found at <https://www.snomed.org/snomed-ct/get-snomed>.

No part of this document may be reproduced or transmitted in any form or by any means, or stored in any kind of retrieval system, except by an Affiliate of SNOMED International in accordance with the SNOMED CT® Affiliate License. Any modification of this document (including without limitation the removal or modification of this notice) is prohibited without the express written permission of SNOMED International.

Any copy of this document that is not obtained directly from SNOMED International [or a Member of SNOMED International] is not controlled by SNOMED International, and may have been modified and may be out of date. Any recipient of this document who has received it by other means is encouraged to obtain a copy directly from SNOMED International [or a Member of SNOMED International. Details of the Members of SNOMED International may be found at <http://www.snomed.org/members/>].

## Table Of Contents

- [Introduction](#)(see page 5)
  - [Background](#)(see page 5)
  - [Purpose](#)(see page 5)
  - [Scope](#)(see page 5)

- Audience(see page 5)
- Business case for frequent delivery(see page 7)
  - Perceived benefits(see page 7)
  - Potential Risks / Issues(see page 8)
    - Managing the perception of a drop in quality vs. the benefit of having the ability to fix things more rapidly(see page 8)
    - Daily vs Monthly release schedule(see page 8)
    - Perceived interoperability issues(see page 9)
    - Visibility of "churn"(see page 9)
    - Prioritisation of maintenance (both content and technical)(see page 10)
- Proposed Content Process changes:(see page 11)
  - Process Timelines(see page 11)
  - Content Authoring processes(see page 12)
    - Content Validation(see page 17)
- Proposed International Edition Release changes:(see page 19)
  - Release frequency(see page 19)
  - Release format(see page 19)
  - What's the impact of multiple effective Times in Delta files?(see page 20)
  - Delta file generation tool(see page 23)
    - Format and requirements(see page 23)
    - Business case(see page 24)
  - Automation of Builds(see page 25)
  - Automation of Release Validation(see page 27)
    - Requirements for the Expansion of the Shared Validation Service:(see page 27)
      - Automated Whitelisting(see page 30)
  - Automation of other Documentation and ancillary content(see page 32)
    - Derivative Products:(see page 32)
  - Critical Incident policy(see page 34)
- Communication(see page 36)
  - Summary(see page 36)
- Future considerations(see page 37)
  - Managed Service products(see page 37)



- [Timeline](#)(see page 37)
  - [How will we manage true "Editions" \(like the US Edition\)?](#)(see page 38)
  - [Derivative Release products](#)(see page 38)
  - [Release Validation as a Service](#)(see page 39)
  - [Release Deliverables](#)(see page 39)
-



## **Introduction**

## **Background**

SNOMED CT terminology provides a common language that enables a consistent way of indexing, storing, retrieving, and aggregating clinical data across specialties and sites of care. SNOMED International maintains the SNOMED CT technical design, the content architecture, the SNOMED CT content (includes the concepts table, the descriptions table, the relationships table, a history table, and ICD mappings), and related technical documentation.

For many years SNOMED International have published their International Edition every 6 months, on 31st January and 31st July each year. However, there is now a proposal to move to a more frequent delivery schedule - the business case for this is set out in this document.

## **Purpose**

This document provides a summary of the proposal for the transition to a more frequent delivery schedule. Both pro's and con's will be elaborated, with a proposal outlined in conclusion in order to provide a roadmap for the transition.

## **Scope**

The primary target for this proposal is the SNOMED CT International Edition.

Other SNOMED products, such as Derivatives and Managed Service packages, will be discussed in this proposal but are not currently targets for the initial transition to more frequent delivery.

## **Audience**

The audience will initially be internal stakeholders within SNOMED International, such as the Content, Technical and Release teams. Once signed off internally, the audience will then include National Release Centres, WHO-FIC release centres, vendors of electronic health records, terminology developers and managers who wish to have an understanding of the proposed changes.



## Business case for frequent delivery

### Perceived benefits

1. New branching strategy should result in less down-time required between release cycles, and therefore enable faster turnaround of authoring content.
2. Smaller content projects have the potential to provide an improved time-to-market for important content changes. This could be especially beneficial for, amongst others:
  1. CRS requests
  2. Drugs models
3. Ability to prevent circular dependencies in long projects, by enabling the phasing of projects to prevent in-flight projects becoming dependent on other in-flight content changes.
4. Ability to deliver published terminology to key stakeholders at more frequent intervals:
  - extension management teams
  - mapping teams
  - translation teams (*should allow for incremental translations, implementers would just need to set expectations with users that translations would still remain behind the International Edition*)
  - developers of AI systems
  - **HOWEVER, this should be caveated by saying that there is no guarantee that changes requested to SNOMED CT will be implemented and released within four weeks, simply because we move to monthly release, because:**
    - *requests will still have to prioritised according to International importance and impact*
    - *authoring and development capacity itself is not increasing, simply the availability of more frequent release cycles in which to publish any content that happens to be ready that month.*
5. Minimal impact for those who do not wish to take more frequent releases, as they will still be able to consume releases every 6 months if they desire.



## Potential Risks / Issues

### Managing the perception of a drop in quality vs. the benefit of having the ability to fix things more rapidly

The initial reaction of the community was that:

- there would be no time for Alpha or Beta Releases - so all Members would have to be comfortable with issues being introduced into Production until the next release.
- all issues that normally get tidied up as part of the normal Content Authoring cycle will become public - they will get fixed quickly but in the meantime there could be an impact to the reputation of the quality of SNOMED CT.

However, those stakeholders involved in either SNOMED or NRC release processes are already aware that the benefits outweigh the small risk of any potential drop in quality. This concern can be mitigated by the following:

- firstly, the new ability to be able to introduce fixes much faster (an inherent benefit with frequent delivery),
- secondly the introduction of more robust gateways into the authoring cycle (see below),
- thirdly, the new approach to scoping of Content projects, to ensure that they are always published as a "complete" release candidate (including documentation and other supporting deliverables), and
- finally, the improved automated validation processes will have increased coverage to catch issues before publication - this can be detailed in the Release Notes to reassure users.

The question was whether or not this view was shared by our vendors and affiliates, and other end users. To that end, we published a survey, where the main question asked was whether or not end users expected negative impact from the introduction of frequent delivery: [https://docs.google.com/forms/d/17Rhxc3TrMgPq1InhAm2G6LkGsaN05\\_-TMKr69WRVdc4/edit#response=ACYDBNiHjEGkP7elqWD-a-56WZydEJy1\\_iH11\\_j11vE8jdcglQQ5-opnoRDplSb0Q8wLpsM](https://docs.google.com/forms/d/17Rhxc3TrMgPq1InhAm2G6LkGsaN05_-TMKr69WRVdc4/edit#response=ACYDBNiHjEGkP7elqWD-a-56WZydEJy1_iH11_j11vE8jdcglQQ5-opnoRDplSb0Q8wLpsM)

The result was that, whilst a few expected some additional work on their systems to get ready for the change to Delta files, not a single user expressed a concern over the final quality of the International content.

### Daily vs Monthly release schedule

Monthly has been agreed to be the most sensible initial frequency, because:





- The stakeholders do not currently believe that we have enough content changes to warrant Daily releases, and therefore most Daily releases would in theory be empty.
- Our community are not in a position to consume Daily releases, and cannot be expected to do so for many years to come.
- Monthly is less of a steep transitional change for internal teams + consumers to process.

## Perceived interoperability issues

There were initially some concerns over interoperability issues, with some people taking each monthly release, and others still waiting for every 6 months. However, this is already a problem as some users take each 6 monthly release, whereas many others only update every 1 or even 2 years. So this is up to the end users to manage - if they want to collaborate with another organisation, they simply have to agree to stay in sync with each others' version of SNOMED CT.

When discussed in the Release AG, they agreed that this wasn't a significant concern, because

- the number of components changing per release is still only a very small percent of the total (and will be even less once we move to Monthly releases),
- this is NOT a SNOMED specific issue, it's prevalent in almost all products/services industries throughout the world that uses versioning, and
- it's the responsibility of the entities who wish to collaborate to ensure that they're all using the same baseline version of SNOMED CT.

This won't change at all when moving to Frequent Delivery - in fact it might even get easier, as agreeing on a Monthly release to align to might be easier for people than agreeing on 6 monthly release to use, as some entities might not be able to take such a huge leap to the next 6 monthly release due to the large number of changes, but could more easily move to a release in a few weeks' time. Australia, for example, made the move to Monthly releases some time ago, and confirmed that although they expected some push-back in this area from their users, in fact they had no feedback whatsoever regarding interoperability issues.

## Visibility of "churn"

One issue raised is that as part of the normal course of the 6 month editing cycle, the authors can change the same concepts (and related components) multiple times, sometimes reverting previous changes as a result of other authoring that happens at a later date. This "churn" is currently hidden within the 6 monthly cycles, but could become more visible once we move the monthly releases.

However, when we discussed this with users they confirmed that they are happy that the benefits of Frequent Delivery (including content being available sooner) far outweigh this minor concern. There are instances, however, where this churn might not have a minor impact on the end users - for example where concept change hierarchy and then back again within a few monthly releases. This can only be properly mitigated by thorough project planning to ensure we prevent partially complete changes being implemented. Wherever possible, projects should only be promoted where all major changes (hierarchy changes, etc) are complete, in order to prevent potential impact to end users. We can also mitigate the churn itself by introducing stronger Release gateways into the authoring process, encouraging authors to consider the impact of each change in the context of all other authoring changes before they are published in the next release.



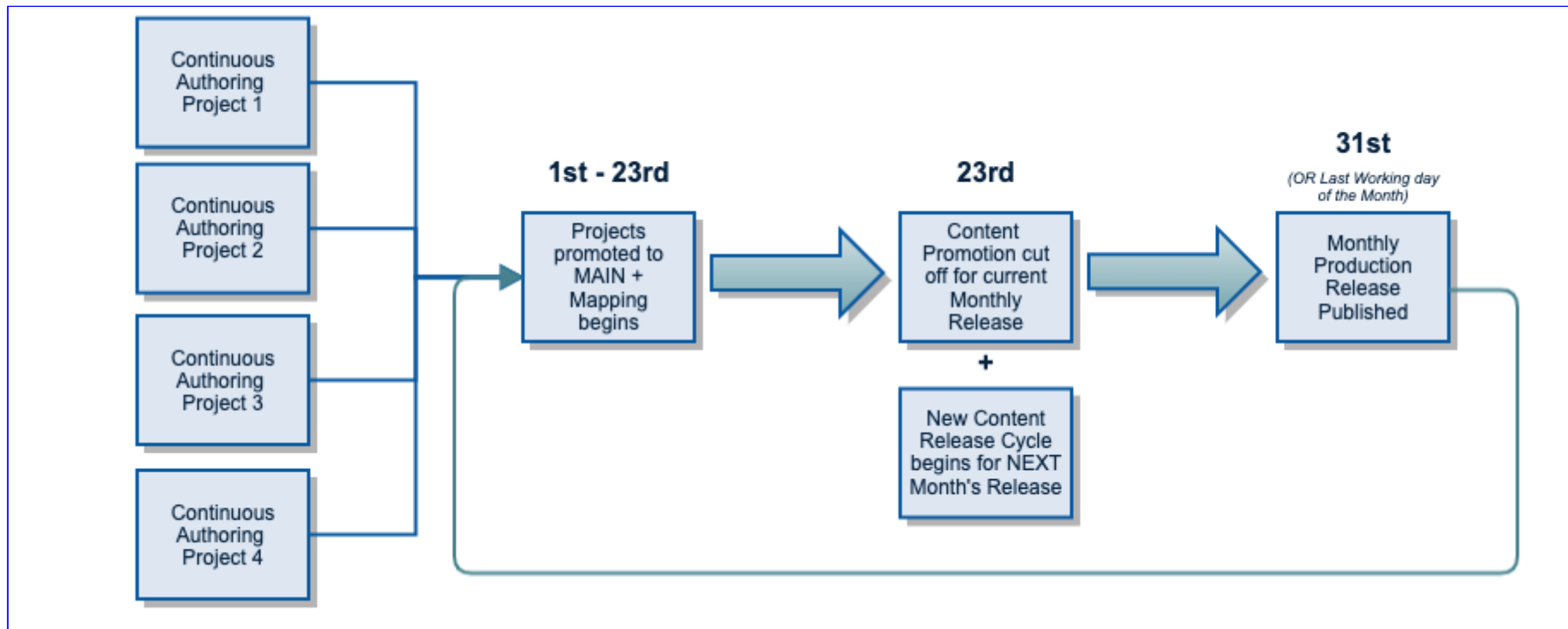
## **Prioritisation of maintenance (both content and technical)**

The prioritisation of maintenance work must now necessarily achieve higher status, both in terms of content fixes for pre-existing content, and technical issues that either prevent efficient authoring or necessitate backend workarounds in order to avoid fixing the real root cause.

If we don't do this, it will result in either a reduction in the quality of SNOMED CT deliverables, or a slowing down of the time to market for content (as the content cannot be promoted to MAIN until all of the validation is successfully signed off). Neither of these situations is desirable, and so we need to all agree that maintenance issues will have to be given significant priority wherever necessary going forward.

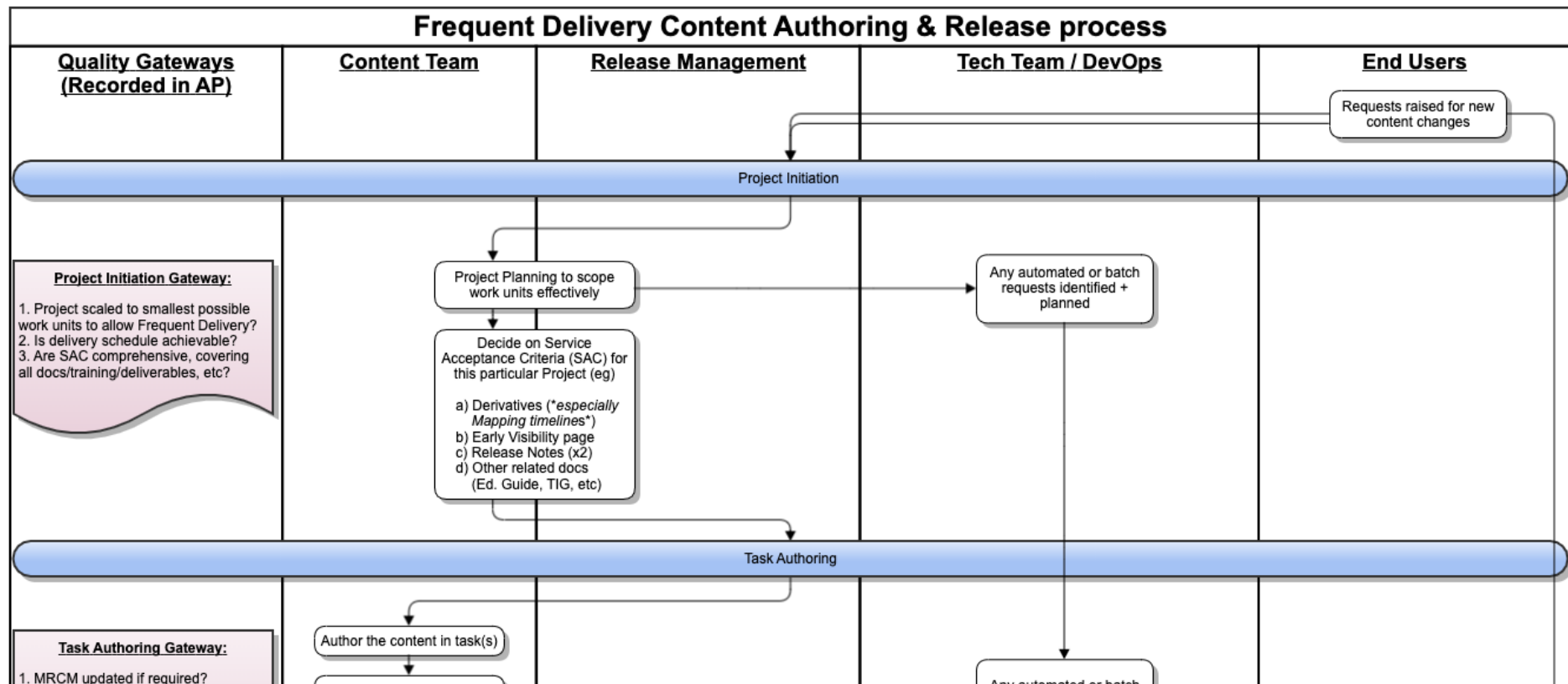
## Proposed Content Process changes:

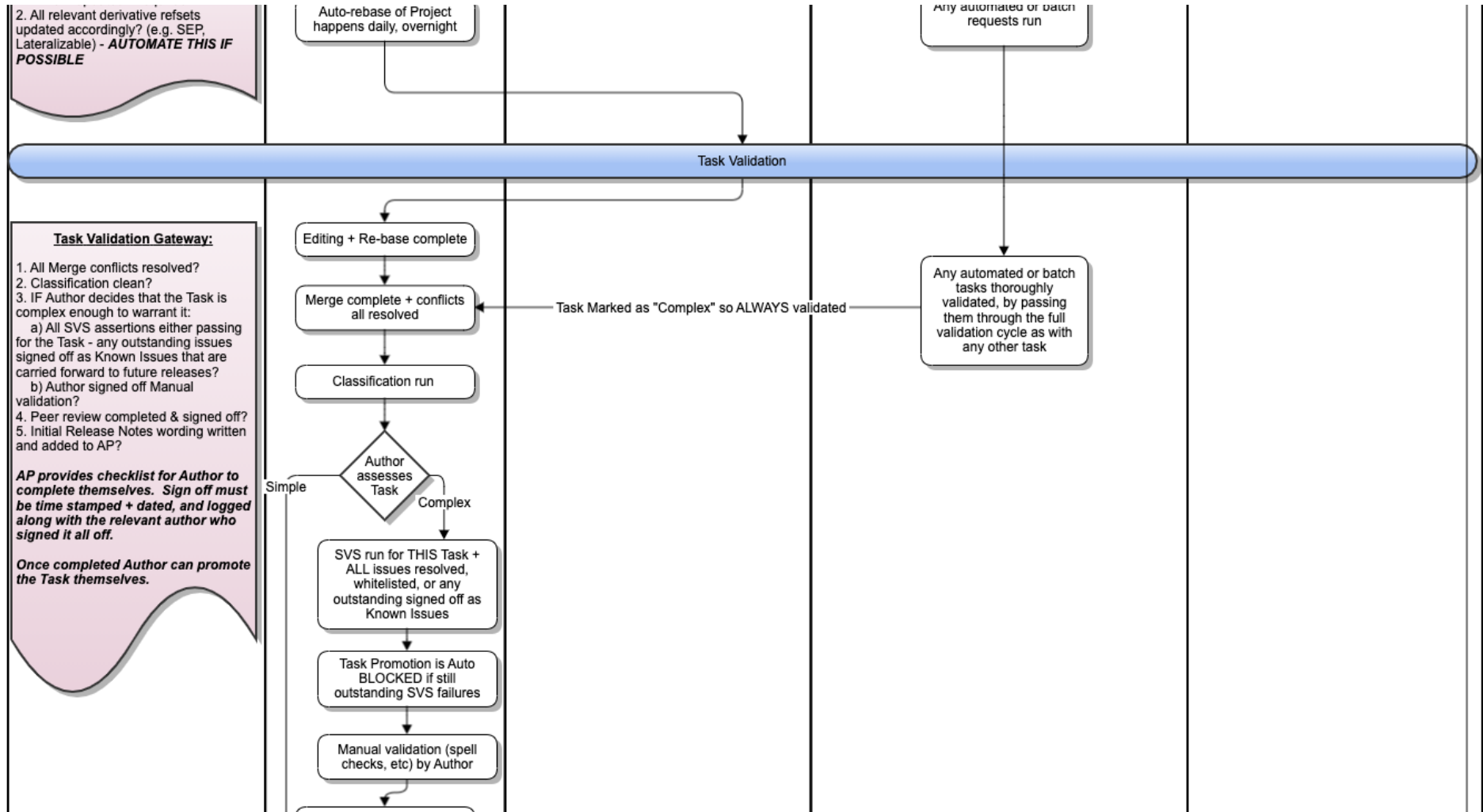
### Process Timelines

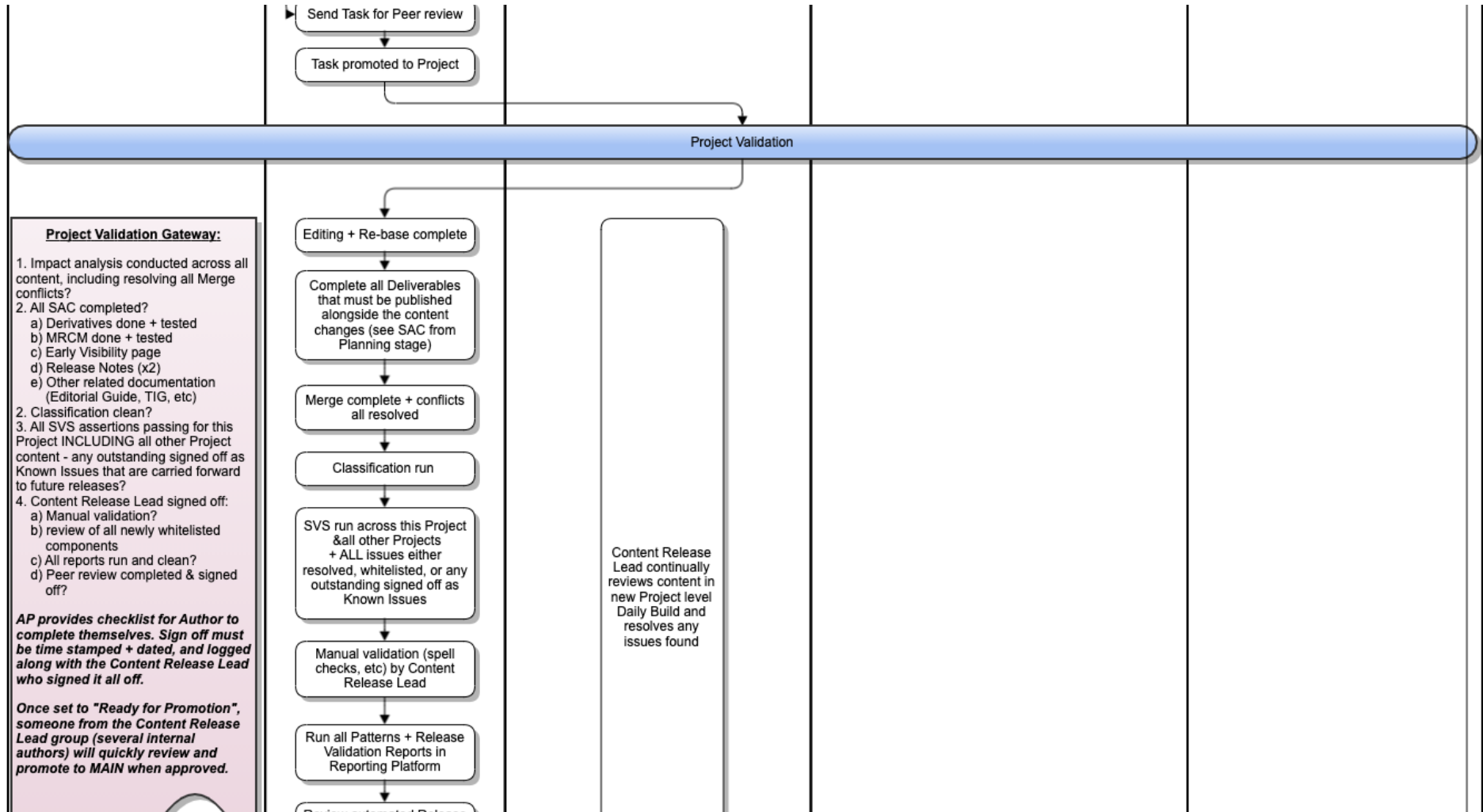


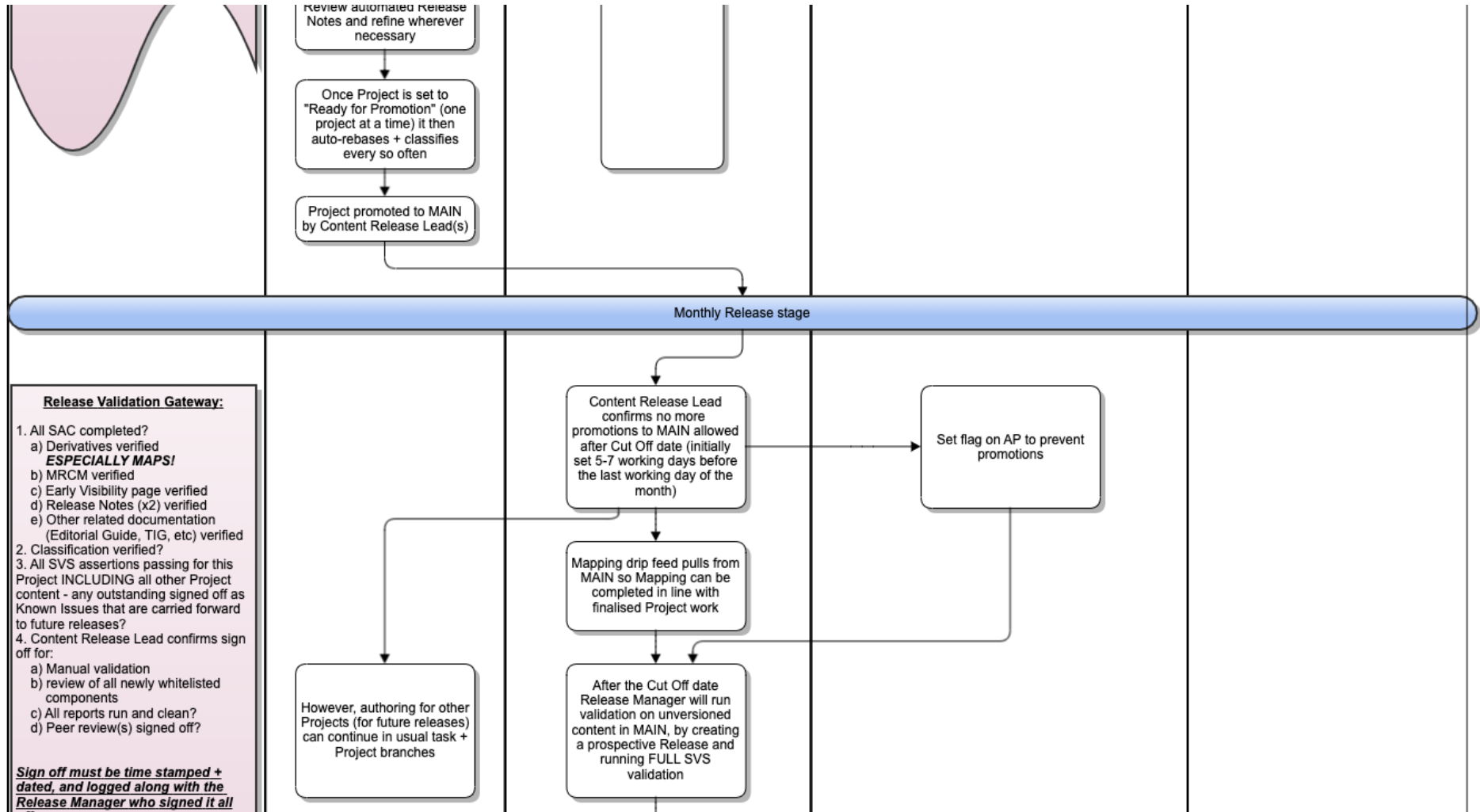
## Content Authoring processes

New Proposed Content Team process workflow:

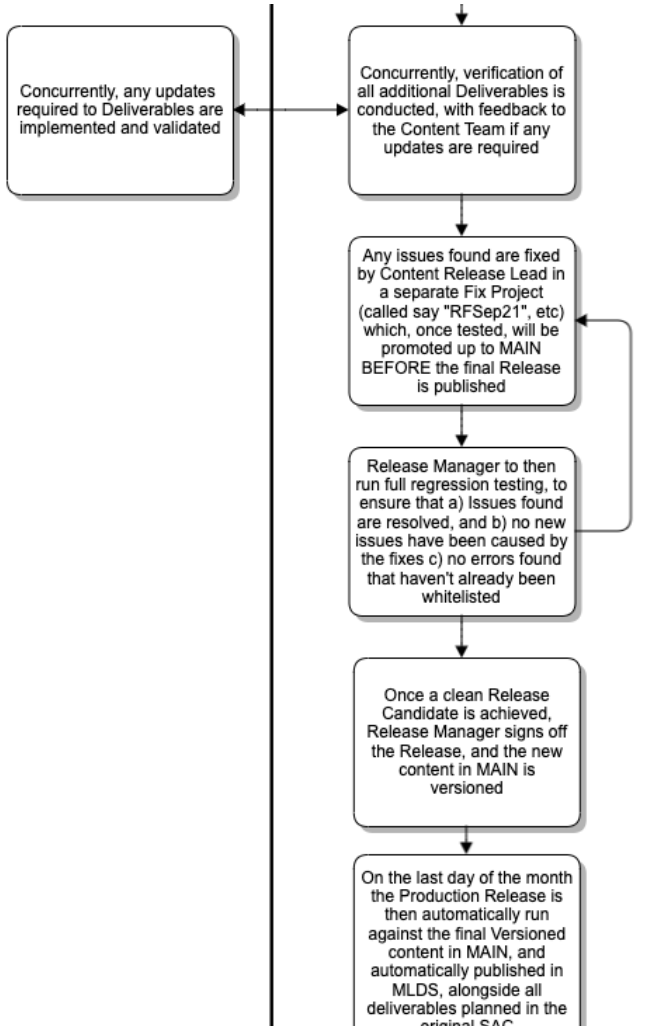




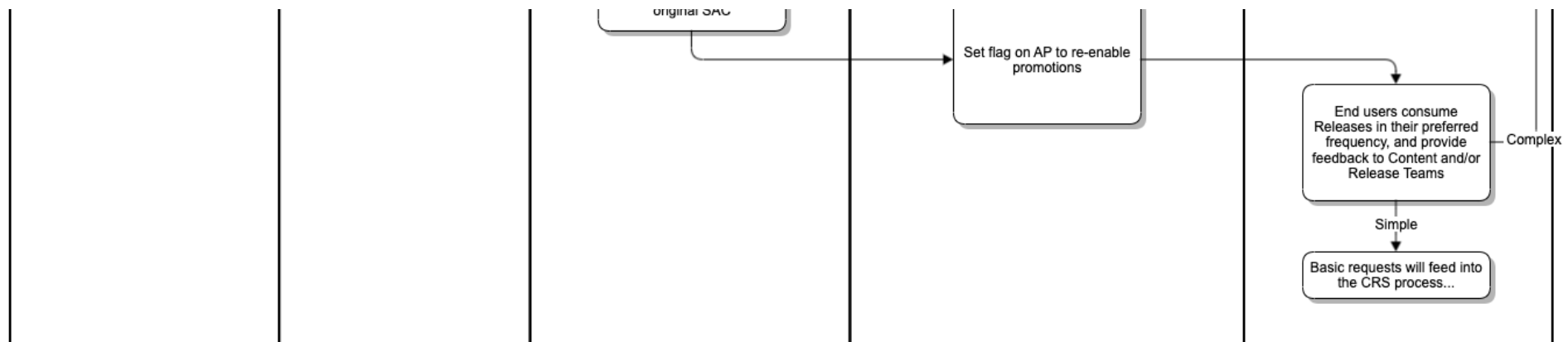




or.  
**Once signed off, the Release Manager will then request Versioning, and the final Release will be automatically Published on the last day of the month, based on the Versioned content only.**







## Content Validation

This needs to be fully automated, covering all know current manual and automated validation routines. This is because the validation will need to be run multiple times per lifecycle of each content change - several times within the editing cycle itself, then again (possibly multiple times) when the content is promoted to MAIN. *Many changes won't necessarily require manual VALIDATION, but do require manual INTERVENTION of the results - (ie) decisions made by authors (not automated) as to how to fix issues presented by the automated validation.*

Thorough analysis of what must be signed off as part of the “done” gateway needs to be conducted for each change.

- This would be a subjective gateway dependent on the size and complexity of the Project.
  - The result should include agreement on the necessary level of validation for each change - for example, any projects considered high risk or to have a high impact, extra peer reviews and even external review prior to promoting to MAIN should be considered.
  - *The Content Team do this now by way of weekly meetings, briefings to CMAG and MF plus CRGs, project groups etc. - so this should not constitute a significant change to the process.*
- **This will then require multiple "levels" of RVF coverage. Perhaps multiple International Edition assertion groups with more/less assertions in them depending on the impact analysis of the change (eg)**

- **International Edition - Basic validation (optional at Task level, as it's too much of an overhead for quick/simple changes)**
- **International Edition - Medium validation (mandatory at Project level)**
- **International Edition - Comprehensive validation (mandatory Full validation at MAIN gateway)**
- However, will an extra few minutes of RVF running time be worth potentially missing some validation issues? Or should we just run ALL assertions for ALL types of change? (don't forget we might also waste time running down false positives for minor changes if we run ALL assertions against each change?)
  - Most people consider that ONLY validations that are pertinent to the content being changed should be run at every stage (ie)
    - Run validation against ONLY task content at task level
      - Same at Project level
      - At MAIN gateway, run ALL validation across ALL content
  - This allows maximum efficiency and prevents duplication of effort - however we will continually assess this as we progress through the first few release cycles, to ensure that there aren't too many issues making it through to MAIN validation that could have more effectively be caught at an earlier stage.
- ***A new feature is required to verify that all authored changes match the final RF2 package changes in the intended way (ie) That the intentions of the authors are being accurately represented in the final content:***
  - **We will use the new Component Stats report which is now aligned with the content changes, so we will automate validation using this report and comparing it to the newly automated Traceability Matrix in the SRS.**
  - **We will also work on expanding the scope of the Component Stats report to ALL components (including derivatives such as refsets and maps)**
  - **The proposed approach is to make this process as efficient as possible by:**
    - **Having an automated report at Project Gateway stage, which shows the Project Lead all of the component changes (*each specific component change, not just change counts as we currently use*), so that they and/or the Content Release Lead can sign each change off, confirming that what they intended to author is actually represented in the changes in the termServer.**
    - **Then, at Release to MAIN Gateway, another automated report is run that**
      - **a) produces a full byte-level comparison between the final RF2 package + the previously published package, and auto-populates the Traceability Matrix**
      - **b) then automatically compares the results to the combination of all of the relevant Component Changed reports (from Project level), and creates a Release Management Exception report that says either: "All projects sign offs match the changes in the RF2 file exactly", or "Here are the (two, or however many) records that don't match, please investigate"**
  - ***This sign off by the Release Manager(s) should then be included in the MAIN gateway checklist, to ensure that this has been completed BEFORE promotion to MAIN***

## Proposed International Edition Release changes:

### Release frequency

After many discussions with the community, it was decided that moving to a daily Release was unnecessary, given that

- a) the Content Team would be very unlikely to have changes ready for Release every day (and therefore many daily releases would either be empty or skipped), and
- b) the end users are not yet ready to consume daily releases (and won't be for the foreseeable future).

In addition to this, it might also prove to be an impediment to interoperability, as keeping different entities aligned with each other's SNOMED versions would potentially be harder with many daily releases than it would be with fewer monthly releases. The only perceived benefit to daily releases would be the ability to get urgent fixes into Production on a daily basis - however as these emergency fixes are few and far between, we can simply manage these through the Critical Incident Policy, and instigate an immediate urgent release on any day inbetween the standard monthly releases.

Therefore, the consensus is that we should ensure that the new Processes support either daily or monthly cycles, and then move to **Monthly** releases initially, evaluating the frequency as we go to see if daily or weekly might help at some point in the future.

The obvious choice of which date to release on each month would be to set it for the last day of the month, as this would allow for the continuation of the traditional 31st January + 31st July releases. However, this may cause some additional work for the automation of the Release builds + validation, as it'll have to take account of effectiveTimes of 28th, 29th, 30th or 31st! So a more preferable option would be to move all releases to the 1st of the month - however we decided that this could have a negative impact on the important message to the community that the transition to Frequent delivery will have no significant impact to anyone who doesn't want to consume monthly releases, as they will have to take 1st Feb + 1st August (instead of 31st January + 31st July), and may have hard coded their import routines, etc.

**So the conclusion was to leave it as the "LAST DAY OF THE MONTH", regardless of the fact that this will be a slightly different date each month.**

**When we have ZERO changes in a month (possible if mapping or other large projects hold up MAIN) we will NOT publish the Monthly release - this prevents wasting time publishing the same release as last month.**

### Release format

Initially it was considered that we would have monthly releases, in addition to 6 monthly roll-up releases on 31st January + 31st July every year. The Delta files for these would be relative to each other, and therefore cause zero impact to the end users who were unable to take monthly releases.



However, everyone is agreed that it's very dangerous to have "different" Release types - for example, Minor monthly releases and only include Major changes in 6 monthly releases. This will defeat the object of Frequent delivery (slowing down the time to market for Major changes), and also obfuscate our message to the community that nothing has to change when we transition because all releases are full RF2 releases and they can therefore just continue to take the Monthly release from 31st Jan + 31st July.

Instead, we categorise all of our change types in advance, and provide (in the JIRA tickets that will comprise the automated Release Notes) a list, in each Release, of the type of changes included in each package. In this way we enable better impact analysis for each user to perform themselves, depending on their own individual use case. This also avoids the pitfall of the first option, which can potentially lead to us classifying the release as major or minor invalidly - as we don't have enough visibility of all of the use cases around across the entire community. Not only that, but even if we did have visibility there are such a wide range of use cases that the same change could easily either be a breaking change or a minor change depending on the end use case!

We may then also need to build some kind of **AUTOMATED** Impact Analysis tool, to allow users to upload their extensions + derivative products and get back at least some level of automated impact analysis against the International Edition package of their selection. This would allow them to "test" their packages against a number of Daily releases to see what potential issues might be relevant to their particular extension/derivative content, in relation to the specific changes in each of those Daily International releases. This will enable them to more effectively decide which Daily Release is "safe" for them to take. - **THIS IS CURRENTLY TARGETED FOR PHASE 2, UNLESS WE GET SIGNIFICANT FEEDBACK TO SUGGEST THAT THIS IS A CRITICAL REQUIREMENT FOR PHASE 1?**

In addition, over the course of the discussions with the community, it has now been decided that Delta files should no longer be published in the International Edition, as there are too many different use cases for the various end users. This means that roll-up releases become irrelevant - **instead we will make each Monthly release equivalent to each other (and dependent on the previous Monthly release), and provide a separate Delta generation tool, allowing end users to create their own Delta files specific to their own needs, depending on which previous release they have consumed.**

## What's the impact of multiple effectiveTimes in Delta files?

- This has been assessed and agreed to be negligible, Australia and US have already implemented this with no effect to users despite initial concerns.
  - We also evaluated the reaction to the July 2020 Rollup Delta package we published - no problems were experienced or concerns raised.
  - Consumers generating their own Delta files with multiple effectiveTimes (covering however many releases they've missed) should therefore cause no foreseeable problems.
  - Snomed International published a small, targeted survey to question whether or not there would be a significant impact to any users
    - No impact identified in survey here: [https://docs.google.com/forms/d/1Rhxc3TrMgPq1InhAm2G6LkGsaN05\\_-TMKr69WRVdc4/edit#responses](https://docs.google.com/forms/d/1Rhxc3TrMgPq1InhAm2G6LkGsaN05_-TMKr69WRVdc4/edit#responses)
    - **Only 23% of respondents used the Delta's, and out of them none identified an issue as long as a Delta service was provided to allow them to continue consuming and publishing Delta's.**

- The issue was discussed at great length with the TRAG ( Multiple effectiveTimes in Delta files - are they required?) - the outcome was as expected:
  - **Delta files generated via the new Delta Generation Tool should include ALL published changes in the interim period in question, NOT just the latest state.**
  - *This is because:*
    - *The question of whether or not the interim states are valid in terms of the Edition was answered, in the sense that even though those states weren't necessarily represented in the Extension content, the Edition necessarily contains the full history of the relevant dependent package(s) as well, and so those interim states were in fact Published states in the dependent package(s). Therefore these interim states are valid parts of Edition packages, and so should be retained accordingly wherever possible.*
    - *The question of churn was also addressed, as a) people consider the likely churn to be relatively insignificant in terms of the size of the overall packages, and b) the Full file was specifically created to record but separate out this churn, so that anyone who needs it can look at the Full files, but anyone who doesn't can simply use the Snapshot files instead.*
    - *The conclusion was therefore that the **COMPOUND DELTA is preferable**, as it allows users to not only calculate a full and correct Full file (which includes not only the Extension's history but also accurately represent the history of all changes to the dependent release package (eg) International Edition), but also to create a less "busy" Snapshot file containing ONLY the latest state of each component, which is what the Atomic delta would have provided anyway.*
  - *Some users proposed an alternative issue:*
    - *We don't yet have specific guidance around this, other than general guidance that relates to this - in the Extensions Practical Guide . For example, we currently say ( on this page ) that in 'File Validation' we check that "For files in the snapshot and delta releases, the id is unique. This would imply that only the most recent state (or row) is included for each id that has changed. However, I can definitely see that this needs discussing, as perhaps someone has a use case for knowing the interim states. We currently teach that installing the 'Full release' is the same as installing an old 'Full release' plus every delta since then - to make this true in this case, you would either need to include both relationship rows in the Delta (breaking the current validation rule of unique ids in the delta), or remove the first (out of date) relationship row from the national edition. From the national edition's perspective this second approach may actually be the correct approach, because if you are doing a historical query over the national edition (e.g. US edition), there was NO national edition in which the relationship was active, so having the Full release of the national edition saying that this relationship was active in the National edition between January 2020 and March 2020, is actually wrong and misleading - and could (in theory) create incorrect historical query results.*
    - *So, in summary, the correct approach would be to keep the validation rule that 'atomic Delta' files (ie the differences between 2 subsequent releases of an edition) should keep the rule that the id must be unique (because anything in between was never actually active in that edition in the stated effectiveTime) .... but for editions that have more frequent releases, a 'compound Delta' file (i.e. one which takes the delta over a period that includes multiple smaller releases) could have multiple versions of the same id (with different effectiveTimes). So, for example, if the international edition moves to daily releases, then the 6-monthly delta will become a 'compound delta', and should include all the*



versions of each id that were published in between the start-date and end-date of that delta. This however, will need to be consolidated in the National Editions, so that they, once again only include each id once in the 'Delta' releases.

- I agree that this is **definitely** something that needs some additional guidance on - both for the scenario you've described, and even more so when we move to Frequent release.
- **Note: If people are pulling down their own Delta's with any possible time frames, then we can't provide Release Notes for this, unless we somehow link up individual Release Note updates to each piece of functionality that gets moved to MAIN, thus allowing it to automatically pull down the relevant Release Note updates for each time period that the Delta is calculated for?**
  - **Is this a concern?**
- Lessons learned from July 2020 Delta Rollup Package:
  - RVF will need to be updated if it is to be accurately run against a Delta rollup package, as:
    - All assertions related to Snapshot/Full files need to be switched off (and replaced where required) as they will not work for a Delta package where ONLY the Delta files are populated
    - Assertions where it checks the Previously Published Package will need to be switched off (and replaced where required), because the previous package is no longer set and defined (as it was in the 6 monthly cycle where it was always the previous July/January package). Instead, the previous package could be anything the user defines when calculating the on demand Delta file. Even this might have worked, except for the fact that the Delta files now contain changes with MULTIPLE effectiveTimes. So, for example, in the July 2020 Rollup Delta it contained changes from both 20200309 + 20200731, but the "Previously Published Package" for the Delta generation was 20200131. Therefore any assertions testing the previously published package validity failed, as the 20200309 changes were not "counted" by the RVF. So, for example, the following assertions failed with false positives:
      - testCategory: "release-type-validation,ICD-MAP,INT,US,GFPF-ICPC2" ,
      - testType: "SQL" ,
      - assertionUuid: "7a961c22-3f51-4f72-9b5e-1dc99c298719" ,
      - assertionText: "There must be actual changes made to previously published extended map refset components in order for them to appear in the current delta." ,
      - failureCount: 3 ,
      - firstNInstances: [{
        - conceptId: "840544004" ,
        - detail: "ExtendedMap: id=c0d288cb-4500-5d6d-90e3-a1b1469ad9bf is in the detla file, but no actual changes made since the previous release." ,
        - conceptId: "840539006" ,



- detail: "ExtendedMap: id=c2c196ac-024b-5f74-a897-dcf09e641564 is in the detla file, but no actual changes made since the previous release."
- conceptId: "840546002" ,
- detail: "ExtendedMap: id=220ab63e-2167-53a6-96c8-cce4521b9191 is in the detla file, but no actual changes made since the previous release."
- *These assertions therefore need to be revised in order to take multiple Previously Published packages into consideration when running them...*
- Some people would also like to see a consistent report (across all national extensions) that allows people to see the high level, human readable changes for each on-demand Delta, rather than just the minute detailed change log... **THIS COULD BE ANSWERED BY DEVELOPING AN EXTENSION TO THE NEW FEATURE FOR IMPACT ANALYSIS OF THE EXTENTION REFSETS etc - VAL-98**

## Delta file generation tool

### Format and requirements

Frequent delivery means that we have to remove the Delta's from the RF2 package and generate the monthly releases with just Snapshot and Full files. However we could also potentially provide a dynamic Delta service to allow anyone to pull down a Delta from any date to any date, in order to allow people to take either a 1 month, 10 months or 3 years Delta, and deal with it in their own comfortable time frame.

For them to be of any use the new Delta file format would need to include the Dependent Release metadata in the Delta file, so that everyone knows what version/effectiveTime the Delta is relative to. We have trialled this in the July 2020 International Edition (and the additional Rollup Delta Release package) - but this only included metadata in terms of the Delta from and Delta to dates (eg)

1. The .JSON file held the following for the official July 2020 release, as the Delta files in this package were relative to the previously published March 2020 release:
  - a. effectiveTime "20200731"
    - i. deltaFromDate "20200309"
    - ii. deltaToDate "20200731"
  - b. Whereas the same metadata file held the following for the July 2020 Rollup Delta release, as the Delta files in this package were relative to the previously published January 2020 release:
    - i. effectiveTime "20200731"
    - ii. deltaFromDate "20200131"
    - iii. deltaToDate "20200731"

## 2. Technical consideration:

- a. *what the latter (which is an example of how the metadata file could work in the future on-demand Delta file packages) did not include was the number of different effectiveTimes that were included in the Rollup Delta package, between the Delta from and Delta to dates. In this case it was only two (20200309 + 20200731) but in theory it could be any number. This data might not be relevant to end users, but we need to ask the question...*
- b. **OPINIONS ON THE OPTIONS HERE?**

### TRAG requirements:

- a) *Delta's to be generated from any point in time to any other point in time*
  - b) *Metadata to be included somehow (to be discussed further in the Metadata Working Group) to record critical information, such as which Dates the Delta is from + to, which Modules are incorporated, etc*
  - c) *Compound Delta's (including ALL changes since the relevant date, including ALL changes in the dependent release package(s), rather than just the latest state - so these are "Full file to Full file" Delta's, as we are used to) are favoured so far, however we should continue to assess any potential use cases for Atomic Delta's (effectively "Snapshot file to Snapshot file" Delta's) as we go along, in case it becomes apparent that there is a valid Business Case to ensure that the new Delta generation tool can provide either or both of these Delta file types...*
  - d) *It needs to support the future requirements for Service Based delivery, once we transition over*
  - e) *Delta at the International level is very simple, but at the Extension level is much more complex due to all of the dependencies, etc. This could also become more involved when we modularise...*
    - *so a Service to download bespoke Delta's (both International and local Extension level) would be helpful to provide to NRC's*
    - *as they need a range of Delta dates PER MODULE, the service would need to be a) clever enough to correctly get the relevant dependencies from all sources, plus b) Validate that the resulting Delta is correct and valid - provide a checksum of some kind (needs to be identified).*
  - f) *We also need to be wary of the fact that there are two different things to be relative to - so you can have a Delta to a release, or a Delta to a date in time, and they can be very different things.*
  - g) *Note: If people are pulling down their own random time frame Delta's, then we can't provide Release Notes for that, unless we somehow link up individual Release Note updates to each piece of functionality that gets moved to MAIN, thus allowing it to automatically pull down the relevant Release Note updates for each time period that the Delta is calculated for?*

## Business case

The Release AG and other members of the community who were consulted do not consider the loss of the Delta files to have any significant predicted impact. Indeed in Australia where this was a major concern initially they experienced zero complaints raised after the transition. As confirmed in the April 2020 TRAG meeting, the entire use case for Delta files is becoming less and less certain - we also have some evidence of usage of Delta's from the TRAG Survey : [https://docs.google.com/forms/d/17Rhxc3TrMgPq1InhAm2G6LkGsaNO5\\_-TMKr69WRVdc4/edit#responses](https://docs.google.com/forms/d/17Rhxc3TrMgPq1InhAm2G6LkGsaNO5_-TMKr69WRVdc4/edit#responses). This is because fewer and fewer users are utilising Delta's to actually upload content, but





instead simply to quickly and cleanly ascertain the latest changes between the previous and current releases. This can more easily be achieved using Release Notes or other means such as a Diff Tool:

- This Diff tool would allow the user to enter the two releases they want to see the differences between, and then
  - just show the Diff between 2x snapshots
    - We could also parameterise whether or not they want to see ALL changes between each dates or just the LATEST change... (so whether it would be like a true delta, or just a snapshot diff with the latest change)
    - We could also potentially include information for WHAT components were changed to and why (ie) historical associations - (eg) this component was inactivated for this reason, and was REPLACED BY this other component...


Having said all that even when Australia offered up a Delta generation tool (multiple times), they were told that it wouldn't be of any use to their users!

So there remains the possibility that this Delta generation tool will not be required - we will need to do several rounds of consultation with the community on Frequent delivery anyway, so we will make the question of whether or not this tool would be useful a part of those discussions...

**TRAG discussed at length on 06/10/2020 and agreed that the requirements are still there, so this will need to be developed (see requirements above) - it's therefore planned for Phase 1**

## Automation of Builds

- We need to automate the build of all currently manually curated content (eg)
  - ModuleDependency Refset files - can be automated
  - RefsetDescriptor files - can't be derived from the content/new refset, so will need to retain this manual step (creation + upload into the termServer)
  - Description Type - can't be derived from the content/new refset, so will need to retain this manual step (creation + upload into the termServer)
  - Identifier - no longer maintained so can be ignored for now (get SRS to assume blank delta in all cases)
- We need to automate the collection, validation and build of all externally maintained refsets (eg)
  - **MRCM x4 - No need for any actions here as the MRCM tool already writes direct to termServer, which exports straight to SRS.**
  - **Lateralizable**
  - **SE**
  - **SP**
  - **(see Peripheral Content section for details...)**
- We need to automate the collection, validation and build of all externally maintained maps (eg)

- ICD-0 - need to ask WCI for current process, to see if it could be automated to write maps straight into termServer/S3? Also speak to WCI about how easy/problematic it would be to move the Mapping Tool drip feed to pull from the Project level instead of MAIN, as otherwise Projects can't bring the map changes along with the content work as a package, ready to promote all together to MAIN.
  - WCI answer - in order to pull data from Projects, we'd have to stand up different Daily Builds from the various projects. This would be risky and fraught with issues with demoting maps, etc
  - So **the alternate** proposal would be to leave all maps to MAIN, which would ensure that they only map against finished content, BUT may delay projects (especially those with large mapping efforts)
- ICD-10 - need to ask WCI for current process, to see if it could be automated to write maps straight into termServer/S3? Also speak to WCI about how easy/problematic it would be to move the Mapping Tool drip feed to pull from the Project level instead of MAIN, as otherwise Projects can't bring the map changes along with the content work as a package, ready to promote all together to MAIN.
  - WCI answer - in order to pull data from Projects, we'd have to stand up different Daily Builds from the various projects. This would be risky and fraught with issues with demoting maps, etc
  - So the alternate proposal would be to leave all maps to the MAIN branch, which would ensure that they only map against finished content, BUT may delay projects (especially those with large mapping efforts)
- ICD-11 - need to ask WCI for current process, to see if it could be automated to write maps straight into termServer/S3? Also speak to WCI about how easy/problematic it would be to move the Mapping Tool drip feed to pull from the Project level instead of MAIN, as otherwise Projects can't bring the map changes along with the content work as a package, ready to promote all together to MAIN.
  - WCI answer - in order to pull data from Projects, we'd have to stand up different Daily Builds from the various projects. This would be risky and fraught with issues with demoting maps, etc
  - So the alternate proposal would be to leave all maps to the MAIN branch, which would ensure that they only map against finished content, BUT may delay projects (especially those with large mapping efforts)
- As we have confirmed that it will not be possible (within the next 12 months) to refine the mapping + refset tools to write directly to SnowStorm (so that we remove all externallyMaintained refsets and maps and pull them all from the termServer) *we need to plan alternative solution for now, ensuring that we can, for example, pull in all maps/refsets into SnowStorm from their external sources before Release, and then just release from SnowStorm...*
- *Ensure there is no downtime in the Authoring Tool (using tagging, etc) whenever we cut and Version/Branch a new Release. We therefore need to ensure that the Authoring Tool + termServer can be available 24/7 once Frequent Delivery is in place (barring code releases for functionality improvements, etc).*
  - This is already possible with current system, as whilst we're versioning MAIN the authors can continue authoring on their Projects
  - *We need to also consider the Daily Build here, in terms of whether or not we still need down time for the Daily Build whilst verisoning etc. Or are we happy with it reverting to blank Delta's once versioning is complete?*
-  VAL-57 - Release Automation **BACKLOG**



- **LOGGING needs to be improved for builds and validation - whether we end up using Jenkins/Rundeck or anything else to automate the builds via SRS, we need MUCH better error handling than we currently have:**
  - **Improved with the introduction of the RAD + DataDog**
  - *Specifics of which system (snowstorm or SRS) to plan the automation of both release builds and validation:*

Looks like keeping the SRS separate from SnowStorm and using that to automate everything is the best solution, because:

    - a) Good to be able to validate things that are separate from SnowStorm (maps etc) - also to allow external users to validate their own SNOMED packages (eventually)
    - b) Also seems to be no real benefit to build the automation into SnowStorm, other than the ability to run DROOLS across the entire terminology

## Automation of Release Validation

### Requirements for the Expansion of the Shared Validation Service:

- All current Patterns Reports in Reporting Platform (as requested by the Content Team) - could be called automatically by the RVF itself and then just display the existing Report(s)
- All Release Validation Reports in Reporting Platform (as requested by the Content Team) - could be called automatically by the RVF itself and then just display the existing Report(s)
- All existing Template validation reports in AP/Reporting Platform should also be included in RVF
- Change the MRCM flag (to run MRCM validation) to be assertion based (ie) decision of whether to run MRCM validation is made at assertion level instead of with SRS config flag...
- Therefore NRC's like Australia will promote all International related content to the core RVF, and only retain and run validation that is local to themselves.
  - This would mean that whenever they identify a new issue, they can simply promote the new test up to us and we can run it and replicate the issue for ourselves, and therefore fix it quickly.
  - It will also share the burden of maintaining the validation rules.
  - Improve documentation for external users to start using the SVS + research putting RBAC in place in order to allow external users to use our Prod RVF itself??
- We need to setup a framework for the Shared Validation Service, to allow us to a) Define the set of assertions that are valid across all content for all users + b) to allow the various NRC's etc to feedback on new/updated assertions on a regular basis to keep this relevant and up to date without a high maintenance overhead for SI

- To achieve this we need to separate the SQL rules from the App in the RVF repo, in order to enable this
- Different levels of assertion groups to be setup for the various new Gateways - small ones for authors to run themselves at task level, then large groups to be run before promoting to MAIN
- In addition, we may also need the ability to merge different unpublished projects, and THEN merge that into the MAIN branch and run impact validation.
- Improve the RVF report to make it more human readable (HTML instead of JSON, etc)
- Covering off the Content Team's current manual validation: [https://docs.google.com/spreadsheets/d/1XetByCBkSJjU85rgpklsVfgd\\_nkgSpTxfwwg6OJbsLw/edit?ts=5e46b885#gid=0](https://docs.google.com/spreadsheets/d/1XetByCBkSJjU85rgpklsVfgd_nkgSpTxfwwg6OJbsLw/edit?ts=5e46b885#gid=0)
- **AutomatedWhitelisting**(see page 30)
  - - per Concept, per Assertion, per SRS Product (as some concepts can be whitelisted for Sweden, say, but not for International).
  - Also need to automatically remove the whitelisting of a concept whenever the concept is updated
  - Report itself should have a separate section of whitelisted failures (and who's signed off the whitelisting of each concept), so we can verify it where required
  - We need a whitelisting refset to manage all the rules for this, and include this in the RF2 packages (so the whitelisting rules can be versioned in the same way as MRCM etc)
  - Page on RAD UI to manage this
- Take the opportunity to refactor the AP validation job handling to go direct from the SVS to Snowstorm, instead of using the outdated Orch Service (as it currently does) - important given the new focus on Task level validation
- **We also need to include the Release Manual validation that still occurs in the International cycle - including:**
  - **Creation of the Content Validation Report for the Content Team (or inclusion of these checks in DROOLS/RVF in the authoring tool)**
    - "VALIDATION OF FINAL INTERNATIONAL RELEASE FILES (Beta release build - July 2020).sql"
    - All manual checks in CONTINUOUS DELIVERY requirements.pdf
  - This includes somehow covering all the eyeballing we currently do (lots of issues found just by human review, which wouldn't be possible in timelines for monthly releases) - eg. issues like ISRS-320 and ISRS-322 that were missed in Jan 2018 - using Natural Language Processing?
    - Or do we just accept these as things that will make it into Production occasionally and have to be fixed on the fly?
- The content team need to confirm whether or not we need improved granularity on "lost descendant" report...
- **ADD A PERFORMANCE EPIC INTO THE SHARED VALIDATION WORK IN ORDER TO MAKE IMPROVEMENTS TO THE RUNNING TIME BEFORE WE**
  - **ADD MORE ASSERTIONS WHICH WILL INCREASE RUNNING TIME**
  - **OPEN UP THE SHARED VALIDATION TO EXTERNAL USERS**
- Proper scoping of all RVF assertion groups per product, especially for MS products - **PLANNED FOR PHASE 2**
- **BULK IMPORTS:**
  - *Currently there is a review of proposed changes, then some spot checks before they get promoted*

- ***In addition then, it would be great to get:***
  - ***Full SVS validation run against each upload***
  - ***For the Bulk imports to be promoted to a separate PROJECT FIRST, before being promoted to MAIN, etc***
- We need the ability to classify other people's extensions against the latest changes to ensure impact analysis is complete down the chain (as far as possible) for major modelling changes at least... **PHASE 2**
- ***Automated Implementation tests of some kind, in order to capture (in advance) any issues that normally wouldn't get found until implementers run the release through their normal processes. This will require significant collaboration with the implementers - NRC's to help collaborate?***
  - ***Covering in SVS working group:***
    - ***Upload validation for systems such as Ontoserver***
      - ***Refset maintenance (Matt)***
      - ***VALIDATION PROJECT WITH CSIRO NOW IN FLIGHT: "Release Assertions" project***
- Need automation of the QA system itself - so some quick but very reliable way to validate RVF + DROOLS Assertions, both old + especially new!
  - ***DROOLS have their own unit tests to cover self-validation of the assertions themselves.***
  - ***RVF therefore needs to either create their own unit tests - OR we need to move all assertions to DROOLS. However the Dev Team have confirmed there are many assertions that DROOLS will never be capable of running, due to technical limitations.***
    - ***Perhaps create a "failure package" that we maintain with a fail point for every assertion - Michael already started one, so we need to estimate how much to bring this up to date and then maintain it going forward (make it part of the critical dev delivery for any new assertion, including those promoted by NRC's, etc)***
- It has been agreed that an exercise will be run by the Dev team to remove the filtering from various RVF assertions, which currently only displays errors from this cycle's content (and therefore hides all historical failures).
  - ***This way we can get an idea of how many concepts we're talking about, across how many different assertions.***
  - ***This should then help us to decide whether or not a "time limit" on whitelisting would be useful - for example, if you whitelist something indefinitely it may never get fixed, so if we allow the authors to set a reasonable time limit on whitelisting (6/12/18 months) then it'll prompt people to take another look at the issue every so often in case it can be more easily fixed in future. However, this is only practical if they won't get thousands and thousands of records re-appearing every few months as a result, and slowing down authoring!***
- We need to identify some way to automate the comparison between the intentions of the authors when making changes, vs the actual changes implemented in the final release.
  - ***This doesn't necessarily have to be Release validation though, could be part of the gateways between authoring + MAIN.***
  - ***Firstly automate whether or not every RF2 Delta change was intentionally authored as part of a valid task in snowstorm***
    - ***Has the task been promoted, validated, classified first***
  - ***Already asserting that what's in the previous full + the new Delta is the same as the current Full, however***

- Also need to verify that there's nothing MISSING from the previous Full, etc
- Also add into the Project Peer Review we should have a list of components + refset members etc listed out, for the Peer Reviewer to check that the component changes (volume, type, etc) are valid and intentional
  - We could also try to do something from the Daily Build point as well, to keep an ongoing eye on this and ensure that, for example, all component changes have tasks related to them where they were intentionally authored, plus whether or not they have been signed off already, etc

## Automated Whitelisting

- This needs to be developed per COMPONENT (Concept, Description, Relationship, etc), per Assertion, per SRS Product (as some concepts can be whitelisted for Sweden, say, but not for International).
  - **ALSO, the content team have requested the ability to set the whitelisting by HIERARCHY as well**
    - **For example, the failures in DROOLS assertion "testType: "DROOL\_RULES", assertionText: "Synonym is preferred in the en-gb refset but refers to a word that has en-us spelling: pressurized" come back failing "Pressurized" when spelt with a Z - this is acceptable in the Product hierarchy (and so can be whitelisted in this hierarchy alone), but needs to continue to fail in all other hierarchies**
  - However we'll also need to allow the authors to say, whitelist a concept for a particular assertion for ALL products, just to save time. Although the same can't be said for Assertions, as if we get to a point where we want to whitelist an entire Assertion then it should just be removed from the relevant Assertion Group completely.
  - Once they have been whitelisted, components should disappear from the RVF reports (for a certain amount of time, or indefinitely??)
  - The authors should then be PROMPTED if any components they've updated in their project have been previously whitelisted, and if so would they like to re-whitelist them?
    - 2 layers of this -
      - soft whitelisted concepts - that need to be fixed at some point in the future, so still need to be visible on th RVF report BUT flagged as whitelisted somehow
      - hard permanently whitelisted concepts - the CAN'T be fixed in future, so should just be hidden in all RVF reports
  - Report itself should have a separate section of whitelisted failures (and who's signed off the whitelisting of each concept + when), so we can verify it where required - defaulted to collapsed down though so most people don't care!
  - We need a whitelisting refset to manage all the rules for this, and include this in the RF2 packages (so the whitelisting rules can be versioned in the same way as MRCM etc)

- Authors should be allowed to see ALL whitelisted components (and who whitelisted them and when) that are in their Project, whenever they promote their Project
- Page on RAD UI to manage this
- **Whitelisting API required - this needs to be automated PLUS:**
  - Needs to be reported - so should be automatically linked to the Known Issues in the Release Notes (so all whitelisted concepts are called out as such in the automated Daily Release Notes)
  - **\*\*\* INCLUDING THE REASON FOR WHITELISTING AS ENTERED BY THE AUTHOR**
  - Need some visibility of outstanding requests for change in the Release Notes - so that people can see what they should ignore because the issue has already been identified and that it'll be fixed soon (would a link through to the Early Visibility page be enough, or do we need more detail? - January 2020 Early Visibility Release Notices - Planned changes to upcoming SNOMED International Release packages)
  - ***Need to ensure that we can whitelist PER COMPONENT, PER ASSERTION - as otherwise if we only whitelist per assertion we run the risk of missing valid issues which are buried in amongst the whitelisted concepts - just like we had with the issues with BE and SE in 2018, where additional (unwanted) classification results were included, and although we picked it up we identified the issues as whitelisted because the valid failures were being reported in amongst hundreds of already whitelisted records for the same RVF assertion!***
- Whitelisting functionality should sit in a separate component service in order to cover DROOLS, RVF + MRCM failures all in the same place - the Dev Team are designing a separate service.
  - We will need to categorise the Criticality of failures - this should be stored against each Assertion (RVF, DROOLS, + MRCM):
    - Tier 1 = Too severe to be whitelisted
    - Tier 2 = Severe - can be whitelisted BUT needs peer review amongst content team
    - Tier 3 = Medium - Can be whitelisted, BUT only by a subsection of content team
    - Tier 4 = Low - Can be whitelisted by anyone without review
  - This will then allow us to control whitelisting using RBAC, as certain groups of people should only be allowed to whitelist certain levels of failure.
  - We could then, in theory, say that:
    - Task Level = authors are allowed to whitelist anything that they have authorisation to (depending on the above criticality)
    - Project level = anything whitelisted at Task level that is high enough criticality that requires peer review should then be flagged up by the system at this point, and promotion prevented until sign off for the whitelisting is obtained.



## Automation of other Documentation and ancillary content

Currently we host all of the related content for a Release up on Confluence (eg) Release Notes, Release Schedules, Traceability matrices, any supporting docs (maps, etc) plus any ancillary content depending on the particular release.

We will need to automate this process, and ensure that all of this content can be quickly and accurately replicated each month, with as much automated or detached from the monthly process as possible (eg) with the Early Visibility pages we host them on Confluence but won't necessarily update them exactly in line with each monthly release - they can be updated at any time, whenever the information becomes available.

*This is currently targeted for Phase 2*

## Derivative Products:

How will we manage derivatives that are currently part of the International Edition Release package?

Current agreed proposal:

- *Keeping them part of the International Edition package, and thereby making the completion of the maps (and other derivatives) part of the necessary work to be completed before a unit of content change can be promoted to MAIN as "Done", will have the following negative impacts:*
  - *Affects the "time to market" of important changes, which is one of the key benefits of Frequent delivery. Given that many of these derivatives are maintained by other teams (ICD-10, MRCM, etc) conflicting priorities might result in delays to content changes being promoted to the Release branch. In addition, the tight timelines involved in the new Release process would make it extremely difficult to deliver the derivative content at the high quality that we expect - especially when there's a large portion of mappable content being delivered.*
    - *We therefore need to ensure that we have accurate mapping estimations at Project initiation, so that we can accurately estimated Release dates:*
      - *Perhaps we can use the ECL query (from WCI tool) to understand what's in scope of mapping, and be able to better estimate the work that will be required for mapping at project initiation??*
  - *Pressure put on the mapping team could induce errors which could otherwise be avoided*
  - *The Dev Team confirmed that the Mapping Tool can pull drip feed from ANY branch, but only ONE branch at a time, so we'll have to push Projects to MAIN and then have them held here until the mapping is complete.*
  - *We will therefore have a dependency here on bringing the Mapping Tool up to date with the Refset Tool, in terms of hooking it up directly to the relevant SnowStorm branch (in this case it will have to be MAIN in order to ensure that it's the final content that is mapped against), so that the maps can be immediately mapped based on the NEW unpublished content that's been created but NOT YET promoted to the Release branch.*





- ICD-10 maps in particular might cause issues -
  - Question asked in TRAG was should we remove it from the International Edition, as mappers might not be able to keep up to date when we move to daily! (or even monthly!)
    - YES - some people think we should remove it from the International Edition package, and have it on it's own cycle with the moduleDependency clearly stated
    - BUT - There are some users that would likely want to have ICD-10 completely up to date...
    - Different opinions - some people don't want to take International content UNTIL ICD-10 is completely up to date, others want to push ahead with the International content regardless of whether or not ICD-10 is up to date yet!
- This was discussed further with content and mapping team...
- **CONCLUSION: Maps should be removed from the International Edition, and disconnected from the International Release schedule. Proposal to be discussed with the TRAG and MF and agreed...**
  - See initial discussions here: **URGENT: Separating ICD maps from the International Edition**
  - As there were no objections, we are now proposing this to all users and consumers of SNOMED CT (including but not limited to members of the MF, CMAG, Customer User Group (<https://confluence.ihtsdotools.org/display/USRG>) to ensure that there are no valid objections before we proceed...
- We also, however, need to cover off another potential issue of how to express (programmatically) which version of the **TARGET** map each change is relevant to? So, for example, when the ICD-10 target map gets updated, how does the user know from one Monthly release to the next, that the ICD-10 target map is different?
  - **First solution is to include this in each Monthly Release Notes**
  - **However, in order to make this info machine readable, we'll also need to improve the Metadata .JSON file included in the International Edition release package to include this data.**

[How do we manage refsets?](#)

#### **Dev Requirement:**

We also have a dependency here on allowing BOTH the Refset Tool + the MRCM Maintenance Tool to be connected to the relevant SnowStorm branch (of the users' choice??), so that the refsets can immediately use the NEW unpublished content that's been created but NOT YET promoted to the MAIN branch. This is because we've now agreed that updating the relevant refsets must be part of the gateway that must be passed **BEFORE** we promote any change to the Release branch, and therefore both tools must have access to the correct branch in order to use these new/updated components. This probably shouldn't be the lowest Dev branch (as then you'll be pulling in a load of unvalidated content into the tools and potentially corrupting them), but something between Dev + Release branches where the majority of authoring validation has been completed but not yet been promoted.

#### **GMDN:**

GMDN + other derivatives will continue with the same release schedule for now (either annually or every 6 months), based on whichever previous monthly International Edition is the most relevant at the time.

## Critical Incident policy

- We need to review and refine the Content Team critical incident process in order to ensure that it's fit for purpose in the new world of frequent delivery. This is because if someone say only takes the release every 12 months (or even worse 24 months), and then finds a critical issue in a now 2 year old release, we would currently have to recall and republish 24 releases!
  - *Instead, we need to have agreement from the Community on a “Forward Only” approach, whereby any issues found (even Critical ones) are fixed from the next Release onwards (or possibly in several Releases time if they're low priority issues). Critical issues would simply have to be communicated out, warning everyone NOT to use any previous impacted releases.*
  - For true critical incidents,
    - do proper recall for ALL affected releases +
    - Provide patched versions of those releases OR preferably Flag all those releases in MLDS to state that they're affected by critical incidents and SHOULD NOT BE USED +
    - Fix it going forward ASAP...
  - Need to consider how best to manage the triage of issues that get reported to decide when to fix them, when to publicise, how to push out the fix release (or tell people about the next daily fix etc)....
  - This will require a proper implementation of Service Management, with Change, Incident, Release, etc, as TRAG agreed *ALL three disciplines are critical to controlling the automated processes*
  - TRAG agreed
    - Flag the affected Release(s) (back to the original inclusion of the affected content) as Critically unsafe, and potentially even remove them from the repository of builds that people can access?
    - For true critical incidents,
      - do proper recall for ALL affected releases +
      - Provide patched versions of those releases, +
      - Flag all those releases in MLDS to state that they're affected by critical incidents +
      - Fix it going forward ASAP...
- *We should also retain the log of the process we followed, decisions made and the resulting fixes etc so that people can refer back to the outcome going forward*



## Communication

### Summary

We need consistent and wide-spread communication on all Release issues, to ensure that there are no surprises for any users.

This must involve, but not be limited to:

1. The creation of comprehensive mailing lists at all levels, so that we can automate comms to the relevant stakeholders (or all stakeholders where appropriate):
  - a. NRC's
    - i. Vendors
    - ii. Affiliates
2. The Early Visibility Page should be made more widely known, so that everyone can ensure they are aware of what is coming up in future releases (especially potentially breaking changes)
3. Everybody to sign up to "Watch" the Release Management portal, as this will then automatically send them updates whenever we send announcements on all things related to Release Management, not just the publications of the RF2 packages: <https://confluence.ihtsdotools.org/display/RMT/SNOMED+International+Release+Management+Home>
4. The TRAG requested that we carefully define and communicate out the scope and benefits of frequent delivery to the community, in order to manage everyone's expectations (for example, just because we're moving to monthly releases doesn't mean that changes will suddenly all get released within 4 weeks, as they'll still need prioritisation!)
  - a. This will be addressed by publishing this Proposal document out to the community for review, once it's been agreed and signed off internally.
  - b. **QUESTION FOR TRAG: IS THIS PROPOSAL DOC FAR TOO DETAILED FOR THE COMMUNITY TO REVIEW? SHOULD WE CREATE A MUCH HIGHER LEVEL VERSION TARGETED AT END USERS?**
    - i. **OR, is this a complex enough proposal that we need to make this level of information available to ensure that all users have the detail they require to thoroughly evaluate the proposal?**
  - c. *As part of this, we also need to work on communications to ensure that all users understand that they can just continue to take any Monthly release they want near the usual 31st Jan/31st July, and therefore their processes don't need to change AT ALL until they want to (in fact Frequent Delivery JUST gives them more options, not less!)*

## Future considerations

### Managed Service products

#### Timeline

The full transition is currently planned for a **MINIMUM OF 12 months AFTER** the International transition (and some customers will not be transitioning at all, at least in the foreseeable future).

Once the International Edition is proven and fully switched over to Frequent Delivery, and we have several clean release cycles under our belt, we then need to move the Managed Service over in exactly the same way as above, only with more interactive testing with the parties involved?

For now, then, the only critical consideration is to ensure that the current manual MS processes can continue unaffected **ON THE SAME CODE BASE - full regression testing is therefore required for ALL Managed Service products before we transition to International frequent delivery, to ensure that the same codebase can support both Frequent International Releases + existing MS Releases simultaneously**.

*Full regression needs to include (but not be limited to):*

- *From the Release perspective we can just run the MS releases against the previous release regardless of how long ago it was, and how many dependent International Edition releases have been published since. However, it's possible that the termServer might not like doing a migration after multiple International releases since the last MS migration, as there will be multiple effectiveTimes to deal with. The Dev Team have confirmed that snowStorm should cope with this, but it needs thorough testing.*
- *Ensure that the code works in the same way for MS as it currently does*
- *Ensure that the SAC works for MS in a very automatic and non-invasive way for the MS users, until they're ready to transition fully.*

How do we manage extensions?

- *Again need to decouple them - MDRS will naturally get a lot bigger - also the versioning process internally currently takes a long time + a lot of effort for each upgrade to new International Edition...*
- *End game is to get the Managed Service upgrades to be self-service - so each customer can decide how frequently they want to upgrade to International depending on their individual users' needs... (so the US might stay on 6 monthly upgrades, but Sweden might move to monthly in line with every International Edition, etc)*

## How will we manage true "Editions" (like the US Edition)?

- This is a lot trickier, as the way the SRS currently works is to take the previous "dependency" International Edition package Delta files, and add them to the new Extension (in this case US) Delta files, in order to create "Edition Delta files". The Snapshot files are then created from these Edition Delta's, and finally the Full files are calculated based on the Delta + Snapshot files.
  - HOWEVER, when you then introduce **MULTIPLE** "dependency" International Edition packages (as the US Edition is run every 6 months by law, and therefore will have six International Edition packages to be dependent upon each cycle instead of one), this causes problems for the Delta and full files. This is because the SRS can't currently cope with either multiple effective times in the Delta files, or with incorporating multiple dependency packages into one Edition.
  - Therefore, we will need SRS changes to ensure that:
    - The user can specify multiple (any number, as if we move up to daily release at any point then this will increase) "dependency" International Edition packages, and the SRS will amalgamate ALL Delta files from all these packages, in the correct effectiveTime order, to create valid "Edition" Delta files for the entire cycle.
    - The SRS will then not only ensure that the Delta files are correct, but will also create valid (if much larger!) Full files, which will need to be carefully built using every single change applied within the relevant cycle, in the correct effectiveTime order.
    - The RVF will also need to be updated to:
      - a) Cope gracefully with multiple effectiveTimes
      - b) Properly validate the Full, Delta and Snapshot files against the multiple "dependency" International Edition package Delta files, to ensure that they've been built correctly

## Derivative Release products

The community agreed that Derivative maps should remain on their own, separate Release schedule for now, so that we can continue to release them whenever they are complete and tested. They will therefore simply be dependent on whichever Monthly International Edition package is the most recent at the time they are released and tested.

**THEFORE THE INITIAL PRIORITY HERE IS TO ENSURE THAT THE EXISTING DERIVATIVE PROCESS CAN CONTINUE UNAFFECTED.**

Finally then, once the transitions to both International + Managed service products are completed we can transition the various derivative products over. However this will be more involved due to the necessary input from the community, which will dictate a far more process intensive approach, as we will necessarily need to change their working processes at the same time (in the same way as we had to for the Content Team and the International Edition).



## Release Validation as a Service

1. Improve the RVF build and containerise process, to allow all end users to easily spin up the RVF locally and run their own packages through it - or even better to host it up as a Service:
2. Ascertain the cost of allowing external users to access the SI RVF API and use that instead.
  - a. *Devops confirmed we need some discovery work to understand what the impact of each use would be. We've just had to upgrade the current servers as they were continuously running out of RAM with just one person using it, and they are currently one of our most expensive boxes.*
  - b. *Also worth mentioning is that we do not currently have the ability to rate limit either. We would probably need to setup a managed API gateway service to do that. We have considered this for the browser in the past but just not had the time to invest figuring it out.*

## Release Deliverables

### HRCM Future requirements

1. Ideally we'd like to move the HRCM out of Confluence, and make it a set of similarly formatted pages that have been introduced in the public MRCMMT in July 2020.
2. There's a conversation to be had about the downstream document dependencies in Confluence though, as they'd need to switch from embedding HRCM tables to embedded external web content.  
If that can be done though, the whole HRCM generation process becomes moot, since it would be implicitly available as soon as a new release branch is created on Snowstorm.