VACMS-15533

https://github.com/department-of-veterans-affairs/va.gov-cms/issues/15533

Gap Analysis: Can the existing notifications framework handle needs for PW Q4 goals?

In short, no. The framework isn't meant to be modified much for purposes other than to meet the needs of 6102. Overall, we have two options:

1. Expand the framework to be more flexible with types of notifications, cadences, and entity types - a very large lift (more than 2 sprints)
2. Implement a new framework which meets our criteria - a large lift (at least 2 sprints)

Requirement	**Gap/FC/Other**	**Notes**
Sending email	✅ FC	The notification system can send emails with (mostly) any content, driven by the existing mail sub-system and related templates.
Additional/differing mail delivery frequencies (2 days in advance of the specified date, day of expiration, when content is auto-archived)	❌ GAP	Framework meant to send once per month only. Additional frequencies would require both backend code, and additional Jenkins job(s).
Expiration of content outside of the 6102 directive. (7 day default expiration for Full Width Banners (Full Width Alert [banner]))	❌ GAP	The framework is meant for monthly notifications for content which has expired, according to the rules the framework dictates. No mechanism for defining a different expiration exists. Additionally, if we were to incorporate additional expiration types and timeframes, we could likely compromise the original intent of the existing framework to comply with 6102.
Works with Blocks	❌ GAP	Not a feature of the existing

		framework.
Aging banner expiration	❌ GAP	Existing framework only works with content expiration determined by the ‘field_last_saved_by_an_editor’ value.
Auto-Archiving	❌ GAP	Not a feature of the existing framework
Permanent Archiving	❌ GAP	Not a feature of the existing framework.
Editors can define a custom expiration date.	❌ GAP	Not a feature of the existing framework.
Configuration for when notifications are sent.	❌ GAP	There is no such configuration available in the current framework. The current monthly rule exists only in code.

Discovery Notes

Expand the framework to work with blocks.

When running the monthly process, the existing framework looks for content that is “expired”, or, hasn’t been updated in a year or more. The query involved, getOutdatedContentForSection():

- Returns only Node content.
- Only returns nodes that:
    - Are published
    - Are not exempted by type
    - Have a `field_last_saved_by_an_editor` timestamp a year or more old
    - Match the provided Section (using field_administration)

If we were to add blocks to the existing framework, we would need to:

- Add the ‘field_last_saved_by_an_editor’ to each block type.
    - And populate this field with a value using an update/similar hook.
    - And add a form alter to populate this field OR update the existing alter to expand to any entity type with this field.
- Update the getOutdatedContentForSection to also query for blocks. This would need to be a second query which would then need to append its results to the existing query (entity queries run against a single entity type). Making this more complicated than nodes is the mix of various field names for the same function on blocks–the section is either in the field_owner or field_administration fields.

However, because of the cadence and type of emails we expect to be sending (2 days in advance, day of, on auto-archive) for PW needs, we could not add the three email cadences to the existing framework without also impacting the existing delivery cadence for expired content.

Block count by type:

type VARCHAR	count(*) BIGINT
alert	104
benefit_promo	1
cms_announcement	1
cta_with_link	1
news_promo	1
promo	153

Concerns and Questions

If a block belongs to the same Section as a node, what would be the default outcome of that for the recipient?

How would the UX work with different entity types in a single email? The existing framework sends a link to a dashboard within the email (it doesn't send the expired content to the user, as the length of a given email might be too much for an email system to handle).
Portions of the existing framework assumes Nodes. There could be parts of the workflow that could be drastically/unavoidably affected by adding Blocks, unless we sidecar blocks as a secondary/alternative workflow.

Date mechanism on the Full Width Banners node form

Adding a date field to a single node/other entity is trivial.

Expand Notifications framework to send emails driven by dates on nodes.

To support this, we would need to either:

- Enhance the current framework to be more flexible
- Create a secondary framework that doesn't impact 6102

Both of these options would be a large lift.

Clarifying Notes from Ticket:

Email recipients: Notifications framework currently sends to the "last modified by" Editor.

❌ The 'last modified by'/'author' of the content is never used. The emails are sent to section members who have expiring content:

From Edmund:

> emails are sent to section members. So we check to see if a section's content is outdated and then notify the members of that section that it is outdated. We do not look at the author at all. We did this because people can move/leave and/or new people can join a section.
> …

Erika noted that the Notifications framework uses Flags. We need to work out the exact use, and whether Blocks / Menus can be flagged, or if the framework will need to be expanded to some other trigger.

❌ The notifications framework ***does not*** use flags.

✅ Only content (nodes) is currently supported. Supporting Blocks is a possibility, but has concerns for overlap with the existing process. Namely, there is no existing mechanism to splinter disparate entity types for unique notifications and delivery schedules.

Notifications engine is based on annual content refresh ("web content should be reviewed by web editors once per year", product brief), and batches monthly emails to the "last modified by" Editor on content approaching the one-year since update mark.

✅ The framework queries for content that is older than 1 year, based on the value of the ‘last saved by editor’ field mentioned earlier.

As far as we know, Notifications system only handles Nodes (not blocks, menus, taxonomy terms, etc)

✅ The framework only works with Nodes today. Adding additional entity types to the existing framework would be a large lift.

Auto-archiving content

This is a separate topic from the bigger picture Notifications framework, but addressed here due to the ticket definition.

This project will require CMS Collab Cycle.

Auto-archiving mechanism doesn’t exist in the CMS today. Will require its own discovery and lift.

Permanently archiving content that cannot be re-published is a new paradigm. Will require its own discovery / lift.

These should be ticketed for separate discovery.

Remaining questions:

- How long after creation should the FWB be archived? If it is archived the same day it expires, we would be sending 2 emails on the same day for a given FWB.
- Does editing a FWB cause the 7 day window to reset automatically?
- Is 7 days a hard minimum, and date picker will not allow user to set a future date greater than 7 days?
- Regarding expiring FWB: do we want to do any FE work to ensure the banner is not displayed after the specified expiration date.
    - Yes. FE should not display banners after the expiration date.