Document Management Versioning Strategy

1.0 Background and Overview

Versioning is an important component of content creation and management. Version management is a key component of enterprise content management. The content we work with to do the Bank's work goes through many changes during its lifetime. Bank teams working to develop content need to:

- Create new content and permission others to contribute to its development *(new content versions)*;
- Find the most current working version for the purpose of reviewing, editing, collaborating, and approving *(working version)*;
- Discover all of the edits and comments that comprise a new version when it is in progress in order to decide which to accept and which to reject, in order to reach consensus and discuss changes with the team members who made them *(working version)*;
- Identify the order in which the changes were made – which supercede which *(working version)*;
- Create a new authoritative version which integrates the institutional knowledge and comments of the experts *(approved version)*;
- Discover earlier, approved versions of the content entity *(superceded versions)*.

2.0 Definition of Versions

Version, as it relates to content management, refers to a form or variation on an earlier instance. Version management and control pertains to all kinds of content, including data, documents/reports, communications, publications, and so on. Version management and control includes overall versioning strategy (state- or change-based approach), rules for selecting versions, the retention of versions and the linking/organization of versions. This proposal addresses the overall strategy, the rules for identifying and enumerating versions, and the linking/organization of versions. Advice on the retention of versions should be provided by the World Bank Group Archivist.

3.0 Business Drivers – World Bank's Need for a Versioning Strategy

Simply put, the current approach to versioning is unmanaged. The current approach:

- Leaves much of the versioning activity entirely outside of the content management systems – in productivity tools;
- Relies on users to deliberately assign a version status at the time of profiling – if and when they file the content in the content management system;
- Uses only generic and uncontrolled version status words – draft, final – leaving interpretation open to users
- Addresses versioning loosely and edition not at all -- version and edition are two dimensions of the same concept – multiple 'final' versions which represent

different editions of the same content may exist entirely independent of one another
- There is no way to pull together in a 'version series' view all working and final versions of a document
- There is no way to mark as superceded content which should not longer be used for Bank work, having a more recent and up to date version.
- Version proliferation is likely occurring on desktops, on network drives and in corporate information systems.

In the past, we have not had the capability to support versioning at the content or the system level. As a result, Bank teams must devote extra effort to tracking and managing versions, including:

- Using email to share working versions of content
- Checking and comparing the date properties to determine which is the most recent working version
- Tracking the sender of an emailed document to determine who made edits
- Looking inside a document in hopes that the last person editing turned on the 'track changes' feature – in order to see the edits

4.0 Version Management Options and Strategies

Two version management and control options were considered: (a) single level version management, and (b) multi-level version management. Both approaches support a change-based versioning management strategy. Both approaches assume that all versions which are retained will be linked through metadata. For a discussion of how the actual changes to documents are captured and managed, please refer to the Annotation Strategy.

4.1 Single Level Version Management Strategy

Single-level versioning assigns a new, incrementally calculated version number to teach changed document. Single-level means a single digit designation for version number (i.e. 1, 2, 3, 4, 5, etc.). Each new edit or change promotes a new version of the document, and assigns a new version number.

While single version numbering would allow us to at least distinguish one version from another. However, it does not allow us to identify or distinguish major and minor versions, nor working from final versions. This option is not a practical option for the Bank given all the consultation, review and editing that is involved in the development of content.

Table xx.
Sample Use Case for Proposed Versioning Strategy

Versioning Action	Version
Irma creates a new minor	Version #1

version document to explain company policy	Status: In process current
The next day Susan checks out the document to add in new text	Version #2 Status: In process current
Susan then checks the document back in and saves her edits	Version #3 Status: Formal final draft
Susan sends a copy to IQ team for review and comments	Version #4 Status: Formal final draft
Luisita checks out the document for review and adds a new policy statements	Version #5 Status: In process current
Luisita checks in the document and saves the changes	Version #6 Status: New version of formal final draft
Denise checks out the document to add text to Luisita's new policy statement	Version #7 Status: In process current
Denise checks in the document and saves the new text	Version #8 Status: New version of formal final draft
The ISG SLT reviews the policy statement and decides it is ready for formal approval. Luisita checks the document out, changes deliberately promotes the policy statement.	Version #9 Status: Formal final version

4.2 Multi-Level Version Management Strategy

Multi-level version management supports:

- Uses a syntax which is comprised of both the major and the minor version properties – 3.1 where first digit (e.g. 3) represents the major version, and the second digit (e.g. 1) represents the minor version;
- Uses a consistent numbering strategy which is both human- and machine-triggered:
  - ③ Newly created content objects where there is no pre-existing major version *(enumeration = 0.0)*
  - ③ Designation of Major Versions based on human-initiated approval and status changes *(enumeration = 1.0)*;
  - ③ Designation of Minor Versions based on system check-out and check-in actions which represent working edits and changes *(enumeration = 1.1, 1.2, 1.3;*
  - ③ Superceded version status (human-initiated status changes) *(enumeration = s1.0)*

- Assigns of a unique version number to each version of a document regardless of whether the content is a major version or a minor version;
- Allows editors to designate a final version by assigning a major version number to a document.
- Enables editors and contributors to determine the nature of the significance of the changes made by others, and to distinguish formal, final versions of documents from working or in-progress versions by checking the version enumeration.

Below is a use case showing how the proposed strategy might be used to designate and track versions of a document. Real names and scenarios have been used to make it easier for us to understand how it would be used.

Table xx.
Sample Use Case for Proposed Versioning Strategy

Versioning Action	Version	Version	Version	Version	
Irma creates a new minor version document to explain company policy	Version: 0.1 Status: In process current				
The next day Susan checks out the document to do add in new text	Version: 0.1 Status: In process current	Version 0.2 Status: Checked Out			
Susan then checks the document back in and saves her edits	Version: 0.1 Status: Superceded	Version 0.2 Status: Current			
Susan sends a copy to IQ team for review and comments	Version: 0.1 Status: Superceded	Version 1.0 Status: Current			
Luisita checks out the document for review and adds a new policy statements	Version: 0.1 Status: Superceded	Version 1.0 Status: Current	Version 1.1 Status: In process Checked Out		
Luisita checks in the document and saves the changes	Version: 0.1 Status: Superceded	Version 1.0 Status: Current	Version 1.1 Status: In process Current Version		
Denise checks	Version: 0.1	Version 1.0	Version 1.1	Version 1.2	

out the document to add text to Luisita's new policy statement	Status: Superceded	Status: Current	Status: In process Current Version	Status In Process Checked Out	
Denise checks in the document and saves the new text	Version: 0.1 Status: Superceded	Version 1.0 Status: Current	Version 1.1 Status: In process Current Version	Version 1.2 Status In Process Current	
The ISG SLT reviews the policy statement and decides it is ready for formal approval. Luisita checks the document out, changes deliberately promotes the policy statement.	Version: 0.1 Status: Superceded	Version 1.0 Status: Current	Version 1.1 Status: In process Current Version	Version 1.2 Status In Process Current	Version 2.0 Status Approved

Option 2: Multi-level Version Management is the preferred option, based on the way that Bank teams work together to create content, specifically the strategy identifies and tracks: (a) Major Versions; (b) Minor Versions; (c) Superceded versions;

5.0 Version Management Architecture

5.1 Version Management Components

Metadata Links and references
Configuration of checkin/checkout functionality for multilevel versioning
Version attribute as metadata

5.2 Version Management Functionality and Processes

The Version Management strategy going forward should support….

- Version management support includes:
  - Check-in/check-out capabilities which trigger and tag versions of content;
  - Version Property management – providing a history of the content state as it is checked in and out of an object store;
  - Capability to individually save and maintain each version of content in the content management system;
  - Ability to bind minor versions to a major version;
  - Capability to retain all edits, not only the most recent backup copy of a document, and incrementally designate new minor versions upon check-in;

- Versioning strategy leverages the property values of a document in productivity software, but also records the version properties as persistent metadata in the ECM metadata repository;
- Version Series Management -- Ability to retain all minor versions of a content object until a person deliberately creates an approved, major version;
- Abililty to distinguish between working versions (minor), approved versions (major), and published (publication location).
    - ③ Content may be approved without needing to be published to a specific location;
    - ③ A single approved version may be published in multiple locations;
    - ③ Minor versions may need to be published before they are approved to a secure space.
- Allows Bank teams to promote a minor version to a major version without checking the document in and out (human initiated promotion)
- Allows Bank teams to demote a major version to a minor version without checking the document in and out (human initiated demotion)
- Allows Bank teams to limit/permission access to minor versions;

6.0 Implementing a Version Management Strategy for New ECM Content

There are two versions of the 'To Be' version management strategy: (a) one which describes operational version management for new content created in ECM; and (b) one which brings the current content into alignment – to the extent possible – with the going forward strategy.

6.1 Operationalizing Version Management

6.2 Versioning of Existing Content

It is not practical to have two conflicting versioning strategies in place in a future enterprise content management system. Therefore, we need to have a strategy for versioning existing content that resembles or is at least not too different from the multilevel versioning strategy.

We realize that any retrospective strategy:

- Is likely to be suboptimal in that it will rely on judgment calls and comparisons
- Will aggregate versions of content at a high level
- Will not have the content base to aggregate at minor levels
    - All content is not likely to be filed in IRIS
    - Older versions may have been overwritten depending on the software configuration

In order to retrospectively convert and aggregate versions, we need a process and we will need some technologies (software which semantically compares and reports on the goodness of fit of two or more content objects). We propose the following process for retrospectively versioning content:

Step 1. Convert metadata for content which already has a Metadata Value to the new strategy

- Working within the existing folder structure
  - Query for content that has status values: draft, final
  - Compare the metadata for items that have status values
  - Semantically compare content
- Where version candidates are found, a human will review and authorize the conversion
- Define conversion table for changing the status value to a version number
- Define the exception rules
  - Where there is only one final version, all draft or other versions will be tagged as minor versions
  - Minor version numbers will be assigned sequentially based on date and time stamps for the documents
- As content is migrated to ECM folders from desktops and network drives, it is automatically compared – semantically and metadata - to existing content
  - Where version candidates are found, person

Step 2. Aggregate the incoming metadata for all content versions

- New attribute 'version number' is created for all ECM content and documents
- Version metadata is recorded in the metadata segment for migrated content
- Add standard text that indicates the version number was added after the fact, as part of the ECM migration strategy and that the version number is for version series management purposes only