# White Paper: Markdown-to-YAML Conversion Script

## Problem

Organizations frequently use Markdown (specifically CommonMark and DocFX flavors) for authoring documentation. Markdown content, however, often lacks structured semantic information, making content validation, reuse, and management challenging. Converting Markdown into structured YAML adhering to Schema.org or other structured schemas solves this problem, but manual conversion is tedious and error-prone.

## Solution

The provided Python script automates the conversion of Markdown documents into structured YAML. It achieves this by:

* Parsing Markdown content into semantically meaningful units and components.
* Using external JSON configuration files to identify and map Markdown blocks to specific components and units.
* Producing YAML compliant with a schema, facilitating structured content management and validation.

## Implementation

The Python script implements the conversion through the following main steps: 1. Recursive Markdown Include Resolution:

* Handles Markdown files with nested includes, resolving content from external files recursively. 2. Markdown Parsing and Component Identification:
* Parses Markdown content, identifies distinct block-level HTML elements (components), such as paragraphs, lists, tables, code blocks, alerts, etc.
* Utilizes a JSON configuration (compMapping.json) for regex-based component identification. 3. Unit Identification:
* Groups identified components into logical units (e.g., procedure, concept, alert sections).
* Employs an external JSON configuration (unitMapping.json) to map components to defined units. 4. YAML Serialization:
* Converts identified units and components into a structured YAML format, compliant with a given JSON Schema.

## Modularity

The script’s modular design allows easy updates and maintenance:

* External Component Configuration:
* compMapping.json defines how Markdown elements map to YAML components.
* Changes or additions to component mappings require edits only to this JSON file.
* External Unit Configuration:
* unitMapping.json defines how sets of components map to logical units.
* Adding or updating unit types involves editing only this external file.

This modularity significantly simplifies adjustments without modifying Python code directly.

## How to Define a New Component

To define a new component: 1. Edit compMapping.json:

* Add a new key-value pair:

“compNewComponent”: “your-regex-pattern”

2. Regex Patterns:

Must match the Markdown content structure to capture component content accurately.

3. Save and Run:

The script will immediately recognize and handle the new component during conversion.

## How to Define a New Unit

To define a new unit:

1. Edit unitMapping.json:

* Add a new key with an array of required components:
* “newUnitType”: [“compParagraph”, “compListUnordered”]

2. Logic:

* Each array represents components that must be present to classify a section as this unit type.

3. Save and Run:

* The script will now classify sections containing these components as the new unit type during conversion.

## Conclusion

This Markdown-to-YAML conversion script provides a robust, scalable, and maintainable solution to structured documentation needs. Its modular design, leveraging external configuration files, enables quick adaptation to evolving documentation structures and requirements.