Auto-Validation System - Master Documentation

Project: VBA Excel Auto-Validation Framework
Version: 2.0 (Consolidated Architecture)
Last Updated: 2026-01-16
Status: Active Development - Phase 1

📋 Change Log

[2.1.0] - 2026-01-16 - Table-Based Configuration (Phase 1 - In Progress)

Status: Development

Major Changes:

- ✅ Created AV_Constants module - all magic numbers centralized
- ✅ Created AV_DataAccess module - centralized table access layer
- ✅ Enhanced AV_Core with table-based configuration loading
- ✅ Added ValidationTargets table support (TableName, Enabled, Mode, Key Column)
- ✅ Eliminated dependency on legacy cell references (B3, B4, B5, M1)
- ✅ Added configuration validation with helpful error messages
- ✅ Implemented validation table caching for performance
- ✅ Added EN/FR header lookup support via ENFRHeaderNamesTable

Breaking Changes: None - this is additive, old code still functions

Next Steps:

- Update AV_Engine to use LoadValidationConfig()

- Update validators to use AV_DataAccess functions
- Test with actual data

[2.0.0] - 2026-01-16 - Architecture Consolidation

Status: Complete

Major Changes:

- Consolidated 15-20 modules into 6 core modules
- Split validation logic into Routing (AV_Validators) and Rules (AV_ValidationRules)
- Removed all AV2_ prefix inconsistencies
- Fixed class name references (clsCellFormat, revStatusRef)
- Eliminated duplicate global variable declarations
- Added missing helper functions (AppendUserLog, ValidateSingleRow, etc.)

Impact: Breaking changes for internal code structure, but maintains functional compatibility

Technical Debt Identified:

- Heavy reliance on cell references (B3, B5, M1, etc.) - brittle and not reusable
- Column letters stored in tables ("M", "AE") - breaks when columns inserted
- Magic numbers scattered throughout (r = 6, i = 12, etc.)
- No schema validation for configuration tables
- Inconsistent data access patterns (direct cells, ListObjects, column scanning)

Next Phase: Phase 1 Quick Wins (see Roadmap below)

[1.0.0] - Original Implementation

Status: Deprecated (being refactored)

Architecture:

- 15-20 separate .bas, .cls, and .frm files
- Multiple validation modules per field type
- Extensive use of AV2_ prefixes
- Ambiguous name conflicts
- Hardcoded cell references throughout

🎯 Project Vision & Goals

Primary Objectives

1. **Maintain Functionality**: All existing validations must continue to work
2. **Reduce Redundancy**: Eliminate code duplication, make debugging easier
3. **Improve Performance**: Faster execution, better user experience for reviewers
4. **Enable Reusability**: Design for future use with different data validation scenarios

Secondary Objectives

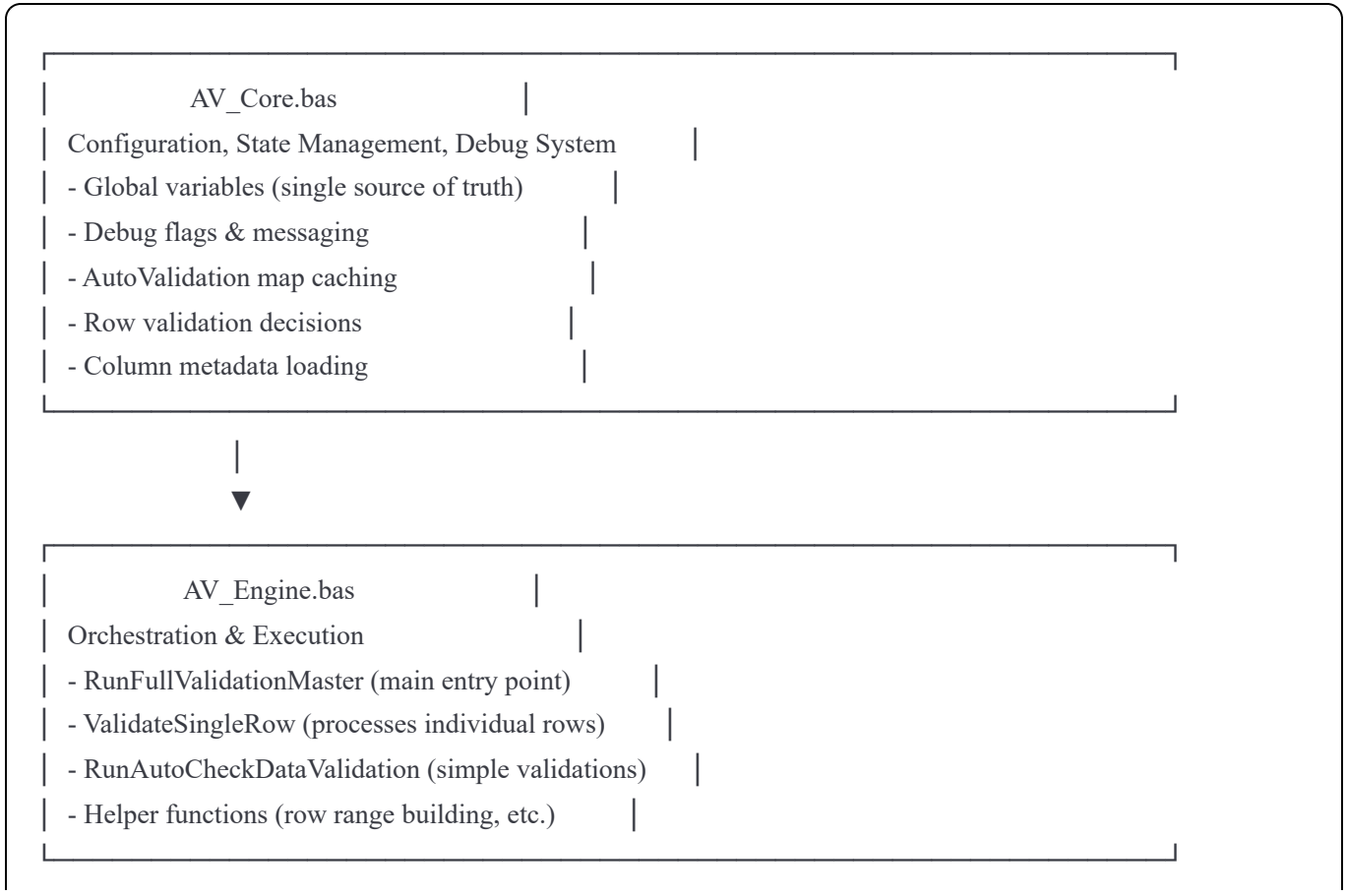
4. **Better Error Handling**: Clear, actionable error messages
5. **Comprehensive Documentation**: Any developer can understand and extend the system
6. **Future Logging Integration**: Prepare for centralized logging system (separate initiative)

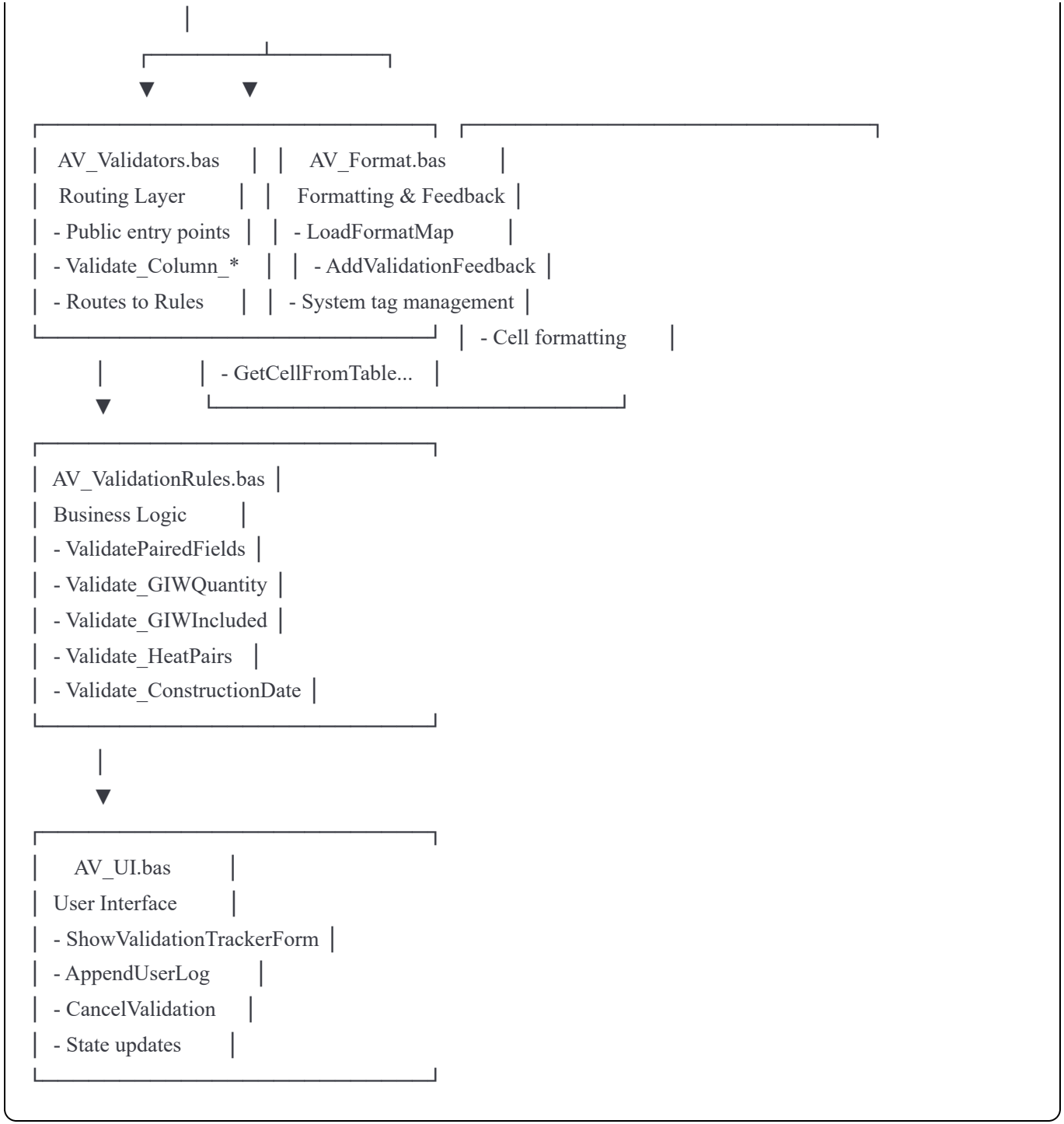
Non-Goals (Current Phase)

- UI redesign for configuration
- Multi-workbook support
- Database integration
- Automated testing framework (future consideration)

🏗️ Current Architecture (v2.0)

Module Structure




AV_Validators.bas
Routing Layer
- Public entry points
- Validate_Column_*
- Routes to Rules
AV_Format.bas
Formatting & Feedback
- LoadFormatMap
- AddValidationFeedback
- System tag management
- Cell formatting
- GetCellFromTable...
AV_ValidationRules.bas
Business Logic
- ValidatePairedFields
- Validate_GIWQuantity
- Validate_GIWIncluded
- Validate_HeatPairs
- Validate_ConstructionDate
AV_UI.bas
User Interface
- ShowValidationTrackerForm
- AppendUserLog
- CancelValidation
- State updates

Supporting Components

Forms:

- `ValidationTrackerForm.frm` - Progress tracking, user log, cancel button

Classes:

- `clsCellFormat.cls` - Cell formatting properties storage
- `revStatusRef.cls` - Review status column references

📊 Configuration Tables Reference

Critical Tables (Required for System Operation)

1. AutoValidationCommentPrefixMappingTable

Purpose: Maps validation functions to columns and configures feedback

Location: Config sheet
Critical: Yes

Columns:

- `Dev Function Names` (String) - Function identifier (e.g., "Electricity")
- `ReviewSheet Column Letter` (String) - Column containing data to validate (e.g., "M")
- `Drop in Column` (String) - Column for validation messages (e.g., "AE")
- `Prefix to message` (String) - English message prefix
- `(FR) Prefix to message` (String) - French message prefix
- `AutoValidate` (Boolean) - TRUE = auto-validate, FALSE = skip

Current Issues:

- Uses column letters (brittle - breaks if columns inserted)
- No validation that referenced columns exist

Future Improvement: Replace column letters with table/column names

---

2. AutoFormatOnFullValidation

Purpose: Defines cell formatting styles for validation results

Location: Config sheet
Critical: Yes

Columns:

- Formatting Key (String) - Format identifier ("Default", "Error", "Autocorrect")
- Autoformatting (Cell Reference) - Sample cell with desired formatting
- KeyFlagPriority (Integer) - Priority for row key formatting (higher = more severe)

Usage:

- Format map loaded once at validation start
- Applied to cells based on validation result
- Row key formatted with highest priority from row

---

3. ForceValidationTable

Purpose: Determines which rows should be validated based on column values

Location: Config sheet
Critical: Yes

Columns:

- `Column` (String) - Column letter to check (e.g., "B")
- `IsBuildingColumnValue` (String) - Value that indicates row should be validated

Logic:

- If ANY row in ForceValidationTable matches, row is validated
- Blank values in both Column and IsBuildingColumnValue = validate if both are blank in data

Current Issues:

- Uses column letters (brittle)
- Logic is "OR" - no way to require multiple conditions

---

4. AutoCheckDataValidationTable

Purpose: Simple dropdown list validations

Location: Config sheet
Critical: Yes

Columns:

- `ReviewSheet Column Letter` (String) - Column to validate
- `Column Name` (String) - English column name
- `Column Name (FR)` (String) - French column name
- `AutoCheck` (Boolean) - TRUE = validate this column

- MenuField Column (EN) (String) - Column in DDM sheet with valid English values
- MenuField Column (FR) (String) - Column in DDM sheet with valid French values
- AutoComment Column (String) - Column for validation messages

Complex Workflow:

1. References DDMFieldsInfo table to find validation data sheet
2. Reads valid values from that sheet's columns
3. Validates against those lists
4. Writes messages to comment column

Validation Rule Tables

5. GIWValidationTable

Purpose: Rules for GIW (Gender-Inclusive Washroom) Included field

Columns:

- GIWIncluded Value (String) - Value in GIW Included field (e.g., "Yes", "No", "Not Applicable")
- Required GIW Quantity Pattern (String) - Expected pattern in GIW Quantity ("0", "1", "#")

Rules:

- "0" = Quantity must be 0,0
- "1" = Quantity must be positive, first ≤ second
- "#" = Quantity must be #,#

Special Auto-correction:

- If Included = "No" and Quantity = "#,#", auto-correct Quantity to "0,0"

---

6. ElectricityPairValidation

Purpose: Valid combinations of Electricity and Electricity Metered fields

Columns:

- `Electricity` (String) - Electricity field value
- `Electricity Metered` (String) - Electricity Metered field value
- `AutoCorrect` (Boolean) - TRUE = apply auto-correction
- `Corrected Electricity` (String) - Corrected value for Electricity
- `Corrected Electricity Metered` (String) - Corrected value for Electricity Metered

Logic: Table-driven pair validation - if exact match found, valid; otherwise error

---

7. PlumbingPairValidation

Purpose: Valid combinations of Plumbing and Water Metered fields

Columns: Same as ElectricityPairValidation

---

8. HeatSourcePairValidation

Purpose: Valid combinations of Heat Source and Heat Metered fields

Columns: Same as ElectricityPairValidation

Special Processing:

- Multi-stage validation with ANY mapping
- Wildcard normalization for "Central Heating Plant - [type]"
- Recursive re-validation after normalization

---

9. HeatSourceANYRefTable

Purpose: Heat source values that map to "ANY" or "ANY(FR)" in validation

Columns:

- Heat Source Value (String) - Heat source that should use ANY mapping

Usage: Stage 2 of heat validation - maps specific values to generic ANY rules

---

Supporting Tables

10. GlobalDebugOptions

Purpose: System-wide debug toggle

Columns:

- Setting (String) - Should be "Global"
- Value (String) - "TRUE" or "FALSE"

Effect: When TRUE, all debug messages print regardless of module settings

---

11. DebugControls

Purpose: Per-module debug flags

Columns:

- `ModuleName` (String) - Name of module
- `Enabled` (String) - "TRUE" or "FALSE"

Effect: Controls debug output for specific modules when GlobalDebugOptions is FALSE

---

12. DDMFieldsInfo

Purpose: Metadata about dropdown menu (DDM) validation data

Columns:

- `Setting` (String) - Setting name
- `Value` (String) - Setting value

Expected Rows:

- ValidationTableName: Name of sheet with valid dropdown values
- StartRowIndex: First data row in that sheet
- EndRowIndex: Last data row in that sheet

---

13. ReviewRefColumnTable

Purpose: Column mappings for review status tracking

Columns:

- `ReviewStatusColumn` (String) - Column letter for review status
- `AutoReviewColumnLetter` (String) - Column letter for auto-review result
- `HumanSetRevStatus` (String) - Column letter for human-set status

Current Issues:

- Table has column headers for column letters (confusing naming)
- Data is stored in first data row, not named properly

---

14. ReviewStatusTable

Purpose: Valid review status values

Usage: Currently referenced but implementation unclear in provided code

---

Legacy/Cell-Based Configuration

Location: Config sheet, loose cells (NOT in tables)

Cell	Contents	Purpose	Issue
B3	Sheet name	Target sheet for validation	Hardcoded reference
B4	Number	Starting row	Hardcoded reference
D4	Number	Row count	Hardcoded reference
B5	Letter	Key column	Hardcoded reference
M1	"English" or "Français"	Language control	Hardcoded reference

Note: These will be migrated to ValidationSettings table in Phase 1

🔄 Data Flow

Validation Execution Sequence

```
1. User Triggers Validation
   └─> RunFullValidation() or RunFullValidationMaster()

2. Initialize
   ├─> ShowValidationTrackerForm (display progress UI)
   ├─> InitDebugFlags (load debug configuration)
   ├─> Set timeout & cancel flags
   └─> Load configuration from Config sheet
      ├─> Target sheet name (B3)
      ├─> Start/end rows (B4, D4)
      ├─> Key column (B5)
      └─> Language (M1)

3. Load Mapping & Format Data
   ├─> GetAutoValidationMap() - validation function mappings
   ├─> LoadFormatMap() - formatting styles
   ├─> GetValidationColumns() - column → function mapping
   └─> GetDDMValidationColumns() - simple validation config

4. Identify Rows to Validate
   ├─> Scan key column for non-empty cells
   ├─> Build array of row numbers
   └─> Filter via ShouldValidateRow() using ForceValidationTable
```

5. Main Validation Loop (for each row)
   ├─> Check cancel flag (user pressed Cancel button)
   ├─> Check timeout (exceeded max time)
   ├─> ValidateSingleRow()
   │   └─> For each mapped validation function
   │   ├─> Check AutoValidate flag
   │   ├─> Get target cell
   │   └─> Application.Run "Validate_Column_[Function]"
   │   ├─> AV_Validators routes to AV_ValidationRules
   │   ├─> Validation logic executes
   │   └─> Calls AddValidationFeedback()
   │   ├─> Looks up message prefix from AutoValMap
   │   ├─> Composes full message
   │   └─> WriteSystemTagToDropColumn()
   │   ├─> Clears old tag for this column
   │   ├─> Applies formatting to source cell
   │   └─> Writes new tag to drop column
   └─> Update progress (every 10 rows)

6. Post-Validation: Simple Dropdown Checks
   └─> RunAutoCheckDataValidation()
       ├─> For each AutoCheck column
       ├─> Validate against DDM valid value lists
       └─> Write errors to comment columns

7. Row Key Formatting
   └─> FormatKeyCell() for each validated row
       ├─> Scan row for formatting
       ├─> Find highest priority format
       └─> Apply to key column cell

8. Cleanup & Completion
   ├─> Re-enable events & screen updating

```
├─> Update form status checkboxes
└─> Display completion message
```

🐛 Known Issues & Technical Debt

Critical Issues

1. Cell Reference Brittleness

- **Severity:** High
- **Impact:** Moving Config sheet cells breaks entire system
- **Locations:** AV_Core, AV_Engine (B3, B4, B5, M1 references)
- **Resolution Plan:** Phase 1, Step 1.1

2. Column Letter Storage

- **Severity:** High
- **Impact:** Inserting columns in target sheet breaks all validations
- **Locations:** All validation mapping tables
- **Resolution Plan:** Phase 1, Step 1.2 (future phase)

3. Magic Numbers

- **Severity:** Medium
- **Impact:** Hard to debug, unclear intent
- **Locations:** Throughout (r = 6, i = 12, ConfigFirstRow = 8, etc.)
- **Resolution Plan:** Phase 1, Step 1.1 (create AV_Constants)

Medium Priority Issues

4. No Schema Validation

- **Severity:** Medium
- **Impact:** Silent failures if table structure wrong
- **Locations:** All table access points
- **Resolution Plan:** Phase 1, Step 1.2

5. Inconsistent Data Access

- **Severity:** Medium
- **Impact:** Code duplication, harder to maintain
- **Locations:** Mix of direct cells, ListObjects, column scanning
- **Resolution Plan:** Phase 2 (create AV_DataAccess layer)

6. No Table Documentation

- **Severity:** Medium
- **Impact:** Developers can't understand table purposes
- **Locations:** N/A - missing entirely
- **Resolution Plan:** This document + TableSchemas sheet

Low Priority Issues

7. Performance: Repeated Table Lookups

- **Severity:** Low (but noticeable with many rows)
- **Impact:** Validation slower than necessary
- **Locations:** GIW, Heat, Pair validations

- **Resolution Plan:** Phase 1, Step 1.4 (table caching)

8. Limited Reusability

- **Severity:** Low (current project works)
- **Impact:** Cannot easily use for other validation scenarios
- **Locations:** Entire architecture
- **Resolution Plan:** Phase 3 (future)

🗺️ Development Roadmap

Phase 1: Quick Wins & Foundation (Current - Week 1-2)

Status: Planning

Goals:

- Eliminate most critical technical debt
- Improve debugging experience
- Boost performance
- Maintain 100% backward compatibility

Tasks:

1.1 Create AV_Constants Module

- Extract all magic numbers
- Document each constant
- Replace throughout codebase

- **Deliverable:** AV_Constants.bas

1.2 Add Configuration Validation

- ValidateConfiguration() function
- Check tables exist
- Check cell values valid
- Helpful error messages
- **Deliverable:** Enhanced AV_Core.bas

1.3 Centralize Config Reading

- ValidationConfig type structure
- LoadValidationConfig() function
- Single source for all config values
- **Deliverable:** Enhanced AV_Core.bas

1.4 Implement Table Caching

- Cache validation tables at start
- Reuse throughout validation run
- Clear cache at end
- **Deliverable:** Enhanced AV_Core.bas with GetValidationTable()

1.5 Create TableSchemas Reference

- New hidden sheet: TableSchemas
- TableSchemaReference table
- Document all tables

- **Deliverable:** Enhanced workbook template

Success Criteria:

- ✅ All validations still work
- ✅ Better error messages
- ✅ 20-50% faster execution (from caching)
- ✅ Developers can find magic number meanings
- ✅ Tables are documented

Phase 2: Standardize Data Access (Week 3-4)

Status: Planned

Goals:

- Consistent API for table operations
- Easier to maintain
- Foundation for future enhancements

Tasks:

2.1 Create AV_DataAccess Module

- GetTableValue() function
- GetTableRow() function
- TableContainsValue() function
- Centralized error handling
- **Deliverable:** AV_DataAccess.bas

2.2 Migrate Existing Code

- Replace direct table access
- Use AV_DataAccess API
- Maintain functionality
- **Deliverable:** Updated AV_* modules

Success Criteria:

- ✅ All table access via AV_DataAccess
- ✅ Consistent error handling
- ✅ No functionality regression

Phase 3: Begin Table-Based Config Migration (Week 5-6)

Status: Planned

Goals:

- Start replacing cell references with tables
- Backward compatible
- Easier to configure

Tasks:

3.1 Create ValidationSettings Table

- Table structure design
- Add to Config sheet

- Optional use (backward compatible)
- **Deliverable:** Enhanced Config template

3.2 Update LoadValidationConfig

- Try table-based first
- Fall back to cell-based
- Transparent to rest of system
- **Deliverable:** Enhanced AV_Core.bas

Success Criteria:

- ✅ New workbooks use tables
- ✅ Old workbooks still work
- ✅ No breaking changes

Phase 4: Reusability Foundation (Future)

Status: Planned (not yet scheduled)

Goals:

- Generic validation engine
- Project-agnostic design
- Reusable validation functions

Tasks: (Details TBD)

- Define validation type enums

- Create generic validators
- Project configuration system
- Validation function registry

👥 Stakeholder Guide

For Reviewers (End Users)

What You Need to Know:

- System works the same way as before
- Same buttons, same form, same process
- May notice validations run slightly faster
- Error messages now more helpful

What Changed:

- Behind the scenes code improvements
- No change to your workflow

For Configurators (Power Users)

What You Need to Know:

- All Config tables still work the same
- New tables being added (optional)
- Better error messages when config is wrong

- TableSchemas sheet documents table purposes

What Changed:

- System now validates config before running
- Clearer error messages if tables missing
- New AV_Constants for magic numbers

Best Practices:

- Don't move cells B3, B4, B5, M1 (yet)
- Use provided table structures
- Refer to TableSchemas for table documentation

For Developers (Extending/Maintaining)

What You Need to Know:

- Architecture now 6 modules (was 15-20)
- All constants in AV_Constants module
- Table access should use AV_DataAccess (Phase 2+)
- Check ValidateConfiguration() for required tables

Code Standards:

- Use constants, never magic numbers
- Access tables via AV_DataAccess when available
- Add table schema to TableSchemas when adding tables
- Comment complex validation logic

Adding New Validations:

1. Add entry to AutoValidationCommentPrefixMappingTable
2. Create Validate_Column_[Name] in AV_Validators
3. Implement logic in AV_ValidationRules
4. Add rule table if needed
5. Update TableSchemas documentation

📚 Reference Materials

Key Code Locations

Global Variables: AV_Core (lines 15-22)
Debug System: AV_Core.InitDebugFlags(), AV_Core.DebugMessage()
Main Entry Point: AV_Engine.RunFullValidationMaster()
Validation Routing: AV_Validators.Validate_Column_*
Business Logic: AV_ValidationRules.*
Feedback System: AV_Format.AddValidationFeedback()

External Dependencies

VBA Libraries Required:

- Microsoft Scripting Runtime (Dictionary support)
- Microsoft VBScript Regular Expressions 5.5 (Regex support)

Excel Features Required:

- ListObjects (Excel Tables)

- UserForms
- Events (Worksheet_Change)

🔐 Version Control

Current Approach: Manual file management

Recommended:

- Git repository for .bas/.cls/.frm files
- Separate config workbook templates
- Tag releases with version numbers

📞 Support & Contacts

Project Lead: [Your Name/Team]
Documentation Maintained By: [Your Name/Team]
Last Review Date: 2026-01-16

Appendix A: Glossary

AutoValidation: Automated field-level validation using custom rules
DDM: Dropdown Menu - simple list-based validations
GIW: Gender-Inclusive Washroom
Review Status: Indicator of whether row needs human review
System Tag: Structured comment format [[SYS_TAG ...]]
Drop Column: Column where validation messages are written

Key Column: Primary identifier column for rows
ForceValidation: Mechanism to filter which rows are validated

---

END OF MASTER DOCUMENTATION

This is a living document. Update the changelog at the top when making significant changes.