**Guide d'Implémentation - Module de Parsing Complet**

**Vue d'Ensemble**

Ce module fournit une solution complète pour analyser et résoudre les références de colonnes dans les expressions de filtrage et spécifications de lecture, avec support avancé pour Excel, CSV et SGBD.

**Architecture**

Le système utilise un **Registry centralisé** (Dictionary) contenant tous les mappings et métadonnées nécessaires pour optimiser l'accès aux données.

**Fonctions d'Entrée Principales**

**1. BuildColumnRegistry() - FONCTION MAÎTRE**

**Signature**

Function BuildColumnRegistry(whatExpression As String, readColumns As String) As Object

**Utilité**

* Point d'entrée unique pour analyser WHAT + READ
* Construit tous les mappings nécessaires en une passe
* Retourne un Registry complet prêt à l'emploi

**Paramètres**

* whatExpression : Expression de filtrage ("@A > 5 AND @C < 10")
* readColumns : Spécification colonnes à lire ("A:E", "[A,C,E]", "A1:B2,EF10:EG10", etc.)

**Exemple d'usage**

Dim registry As Object

Set registry = BuildColumnRegistry("@Nom LIKE 'Dupont\*' AND @Age > 25", "A:F")

**Retour** Registry Dictionary avec toutes les structures :

* WHAT\_FIELDS : Champs dans expression
* READ\_FIELDS : Champs à lire
* ALL\_REQUIRED : Union des deux
* READ\_ORDER : Ordre d'affichage préservé
* COMPARISON\_FIELDS : Champs utilisés dans comparaisons
* Mappings de positions (source/extract)

**2. ResolveExcelDynamicReferences() - RÉSOLUTION DYNAMIQUE**

**Signature**

Function ResolveExcelDynamicReferences(registry As Object, Optional context As ExcelResolutionContext) As Boolean

**Utilité**

* Résout les plages nommées Excel en adresses réelles
* Support tableaux structurés et en-têtes colonnes
* Gestion workbook ouvert/fermé

**Quand l'utiliser**

* Après BuildColumnRegistry() si références nommées détectées
* Avant traitement des données Excel

**Exemple d'usage**

Dim registry As Object

Set registry = BuildColumnRegistry("@Facture > 1000", "Clients:Montants")

If ResolveExcelDynamicReferences(registry) Then

' Références résolues, continuer traitement

Else

' Erreur résolution, gérer fallback

End If

**3. BuildAdvancedPositionMappings() - MAPPINGS AVANCÉS**

**Signature**

Sub BuildAdvancedPositionMappings(registry As Object, sourceDataArray As Variant)

**Utilité**

* Construit mappings positions réelles depuis données source
* Optimise ordre d'extraction pour performance
* Crée index inverses pour accès O(1)

**Quand l'utiliser**

* Après chargement des données source
* Avant traitement/filtrage des données

**Exemple d'usage**

Dim sourceData As Variant

sourceData = Range("A1:Z1000").Value ' Données Excel

Dim registry As Object

Set registry = BuildColumnRegistry("@Prix > 100", "A:E")

' Construire mappings depuis données réelles

BuildAdvancedPositionMappings registry, sourceData

**Fonctions de Consultation (API Publique)**

**Accès aux Colonnes**

**GetWhatColumns(registry) → Object**

Retourne Dictionary des champs dans expression WHAT

**GetReadColumns(registry) → Object**

Retourne Dictionary des champs à lire (optimisé READ\_EQUALS\_WHAT)

**GetAllRequiredColumns(registry) → Object**

Retourne Dictionary union de tous les champs nécessaires

**GetComparisonFields(registry) → Object**

Retourne Dictionary des champs utilisés dans comparaisons

**Exemple d'usage**

Dim whatCols As Object: Set whatCols = GetWhatColumns(registry)

Dim readCols As Object: Set readCols = GetReadColumns(registry)

Debug.Print "Champs WHAT: " & whatCols.Count

Debug.Print "Champs READ: " & readCols.Count

**Mappings de Position**

**GetExtractPosition(registry, fieldRef) → Long**

Position du champ dans tableau extrait ordonné

**GetFieldAtExtractPosition(registry, position) → String**

Champ à la position donnée dans tableau extrait

**GetSourcePosition(registry, fieldRef) → Long**

Position du champ dans données source

**SetSourcePosition(registry, fieldRef, sourcePos)**

Définir position source (appelé lors extraction)

**Exemple d'usage**

' Connaître ordre d'extraction

Dim pos As Long: pos = GetExtractPosition(registry, "@Nom")

Debug.Print "@Nom sera en position " & pos & " dans résultats"

' Mapping inverse

Dim field As String: field = GetFieldAtExtractPosition(registry, 1)

Debug.Print "Première colonne résultat: " & field

**Utilitaires**

**HasColumn(registry, fieldRef) → Boolean**

Teste existence d'un champ dans le registry

**GetColumnCount(registry) → Long**

Nombre total de colonnes requises

**GetColumnIndex(registry, fieldRef) → Long**

Index numérique du champ (A=1, B=2, etc.)

**Fonctions d'Analyse Avancée**

**AnalyzeComparisonContexts(expression) → Collection**

**Utilité** Analyse sophistiquée des comparaisons dans expression

**Retour** Collection de ComparisonContext avec :

* FieldName : Champ comparé
* Operator : Opérateur (=, >, LIKE, IN, etc.)
* ComparedValue : Valeur de comparaison
* ContextType : Type (FILTER, JOIN, SUBQUERY, etc.)

**Exemple d'usage**

Dim contexts As Collection

Set contexts = AnalyzeComparisonContexts("@Age BETWEEN 18 AND 65 AND @Nom LIKE 'Dup\*'")

Dim i As Long

For i = 1 To contexts.Count

Dim ctx As ComparisonContext: ctx = contexts(i)

Debug.Print ctx.FieldName & " " & ctx.Operator & " " & ctx.ComparedValue

Next i

**GetOptimizationRecommendations(registry) → Collection**

**Utilité** Fournit recommandations d'optimisation basées sur l'analyse

**Exemple d'usage**

Dim recommendations As Collection

Set recommendations = GetOptimizationRecommendations(registry)

Dim i As Long

For i = 1 To recommendations.Count

Debug.Print "Recommandation: " & recommendations(i)

Next i

**Workflow d'Implémentation Complet**

**Étape 1 : Analyse Initiale**

' 1. Construire registry depuis spécifications utilisateur

Dim registry As Object

Set registry = BuildColumnRegistry(userWhatExpression, userReadColumns)

' 2. Vérifier besoin résolution dynamique (si Excel)

If HasNamedReferences(registry) Then

If Not ResolveExcelDynamicReferences(registry) Then

' Gérer échec résolution

MsgBox "Impossible de résoudre toutes les références nommées"

End If

End If

**Étape 2 : Chargement Données**

' 3. Charger données source (exemple Excel)

Dim sourceRange As Range

Set sourceRange = Workbooks("MonFichier.xlsx").Worksheets("Données").UsedRange

Dim sourceData As Variant: sourceData = sourceRange.Value

' 4. Construire mappings positions réelles

BuildAdvancedPositionMappings registry, sourceData

**Étape 3 : Optimisation Extraction**

' 5. Obtenir ordre optimal d'accès

Dim optimalOrder As Collection

Set optimalOrder = GetOptimalAccessOrder(registry)

' 6. Préparer tableau résultat dans bon ordre

Dim resultData() As Variant

ReDim resultData(1 To UBound(sourceData, 1), 1 To GetColumnCount(registry))

**Étape 4 : Extraction Sélective**

' 7. Extraire seulement colonnes nécessaires

Dim allRequired As Object: Set allRequired = GetAllRequiredColumns(registry)

Dim sourceCol As Long, extractCol As Long

For Each fieldRef In allRequired.Keys

sourceCol = GetSourcePosition(registry, fieldRef)

extractCol = GetExtractPosition(registry, fieldRef)

' Copier colonne source → position extract

Dim row As Long

For row = 1 To UBound(sourceData, 1)

resultData(row, extractCol) = sourceData(row, sourceCol)

Next row

Next fieldRef

**Gestion des Formats Supportés**

**Formats READ Supportés**

| **Format** | **Exemple** | **Description** |
| --- | --- | --- |
| Range Excel | A:E ou A1:E10 | Range contiguë |
| Multi-ranges | A1:B2,EF10:EG10 | Ranges multiples (ordre préservé) |
| Liste bracket | [A,C,E] | Colonnes spécifiques |
| Numérique | 1:5,8,10:12 | Index numériques |
| Nommé Excel | Clients:Montants | Résolution dynamique |

**Formats WHAT Supportés**

| **Format** | **Exemple** | **Description** |
| --- | --- | --- |
| Comparaisons | @A > 5 AND @B < 10 | Opérateurs standard |
| LIKE | @Nom LIKE 'Dup\*' | Correspondance patterns |
| IN | @Code IN ('A','B','C') | Valeurs multiples |
| BETWEEN | @Age BETWEEN 18 AND 65 | Ranges de valeurs |
| Subqueries | @ID IN (SELECT...) | Requêtes imbriquées |
| Jointures | @ID1 = @ID2 | Comparaisons inter-champs |

**Diagnostics et Debug**

**Mode Debug**

' Activer debug dans configuration

FDXH\_Config("DebugMode") = True

FDXH\_Config("LogParsingSteps") = True

' Diagnostic complet

DiagnoseRegistry registry

**Validation**

' Vérifier cohérence mappings

If Not ValidateMappingConsistency(registry) Then

MsgBox "Incohérence détectée dans les mappings"

End If

' Vérifier résolution complète

If Not AreAllReferencesResolved(registry) Then

MsgBox "Certaines références n'ont pas pu être résolues"

End If

**Bonnes Pratiques**

**Performance**

* Utilisez READ\_EQUALS\_WHAT quand possible (économie mémoire)
* Priorisez champs de comparaison dans ordre extraction
* Validez mappings avant traitement massif

**Robustesse**

* Toujours tester ResolveExcelDynamicReferences() avant usage
* Gérer cas échec résolution avec fallbacks appropriés
* Valider cohérence mappings en mode debug

**Maintenance**

* Utilisez fonctions diagnostics pour identifier optimisations
* Documentez références nommées Excel utilisées
* Testez avec différents formats d'entrée

**Dépendances**

**Modules Requis**

* mod\_Global : Variables globales et types
* FDXH\_Config : Configuration système

**Références Excel**

* Microsoft Excel Object Library (si résolution dynamique)
* Microsoft Scripting Runtime (Dictionary)

**Configuration Minimale**

' Dans mod\_Global

Public FDXH\_Config As Object

' Initialisation

Sub InitializeExtendedConfig()

Set FDXH\_Config = CreateObject("Scripting.Dictionary")

FDXH\_Config("DebugMode") = False

FDXH\_Config("LogParsingSteps") = False

FDXH\_Config("MaxRowsInMemory") = 100000

FDXH\_Config("MaxInValues") = 50

FDXH\_Config("EnableShortCircuit") = True

End Sub

Cette documentation couvre l'ensemble des fonctionnalités. Le module est conçu pour être extensible et maintenir la compatibilité avec votre architecture existante.