

GSI.IH.Common.Translation - Missing Configuration Classes

Configuration Classes Implementation

1. TranslationOptions.cs

Location: src/GSI.IH.Common.Translation/Configuration/TranslationOptions.cs

```
namespace GSI.IH.Common.Translation.Configuration;

/// <summary>
/// Configuration options for the translation engine.
/// </summary>
public sealed class TranslationOptions
{
    /// <summary>
    /// Section name in configuration file.
    /// </summary>
    public const string SectionName = "Translation";

    /// <summary>
    /// Enable caching of translation definitions.
    /// Default: true
    /// </summary>
    public bool EnableCaching { get; set; } = true;

    /// <summary>
    /// Cache duration in minutes.
    /// Default: 10 minutes
    /// </summary>
    public int CacheDurationMinutes { get; set; } = 10;

    /// <summary>
    /// Maximum payload size in kilobytes.
    /// Default: 1024 KB (1 MB)
    /// </summary>
    public int MaxPayloadSizeKb { get; set; } = 1024;

    /// <summary>
    /// Maximum number of field mappings per translation definition.
```

```

    /// Default: 500
    /// </summary>
    public int MaxFieldMappings { get; set; } = 500;

    /// <summary>
    /// Enable detailed logging.
    /// Default: false
    /// </summary>
    public bool EnableDetailedLogging { get; set; } = false;

    /// <summary>
    /// Timeout for transformation operations in seconds.
    /// Default: 30 seconds
    /// </summary>
    public int TransformationTimeoutSeconds { get; set; } = 30;

    /// <summary>
    /// Enable performance metrics collection.
    /// Default: false
    /// </summary>
    public bool EnablePerformanceMetrics { get; set; } = false;

    /// <summary>
    /// Default validation profile to use if not specified.
    /// Default: "Strict"
    /// </summary>
    public string DefaultValidationProfile { get; set; } = "Strict";

    /// <summary>
    /// Enable automatic type conversion.
    /// Default: true
    /// </summary>
    public bool EnableAutoTypeConversion { get; set; } = true;

    /// <summary>
    /// Parallel processing threshold (number of field mappings).
    /// If field mappings exceed this number, use parallel processing.
    /// Default: 100
    /// </summary>
    public int ParallelProcessingThreshold { get; set; } = 100;
}

```

2. ValidationOptions.cs

Location: src/GSI.IH.Common.Translation/Configuration/ValidationOptions.cs

```

namespace GSI.IH.Common.Translation.Configuration;

/// <summary>
/// Configuration options for validation.
/// </summary>
public sealed class ValidationOptions
{
    /// <summary>
    /// Section name in configuration file.
    /// </summary>
    public const string SectionName = "Validation";

    /// <summary>
    /// Enable validation.
    /// Default: true
    /// </summary>
    public bool Enabled { get; set; } = true;

    /// <summary>
    /// Default validation profile.
    /// Options: "Strict", "Relaxed", "None"
    /// Default: "Strict"
    /// </summary>
    public string DefaultProfile { get; set; } = "Strict";

    /// <summary>
    /// Maximum number of errors to collect before stopping (Relaxed mode).
    /// Default: 50
    /// </summary>
    public int MaxErrorCount { get; set; } = 50;

    /// <summary>
    /// Enable JSON schema validation.
    /// Default: true
    /// </summary>
    public bool EnableSchemaValidation { get; set; } = true;

    /// <summary>
    /// Enable mandatory field validation.
    /// Default: true
    /// </summary>
    public bool EnableMandatoryFieldValidation { get; set; } = true;

    /// <summary>
    /// Enable format validation (regex).
    /// Default: true

```

```

    /// </summary>
    public bool EnableFormatValidation { get; set; } = true;

    /// <summary>
    /// Enable range validation.
    /// Default: true
    /// </summary>
    public bool EnableRangeValidation { get; set; } = true;

    /// <summary>
    /// Enable date range validation.
    /// Default: true
    /// </summary>
    public bool EnableDateRangeValidation { get; set; } = true;

    /// <summary>
    /// Treat warnings as errors.
    /// Default: false
    /// </summary>
    public bool TreatWarningsAsErrors { get; set; } = false;

    /// <summary>
    /// Validation timeout in seconds.
    /// Default: 10 seconds
    /// </summary>
    public int ValidationTimeoutSeconds { get; set; } = 10;

    /// <summary>
    /// Custom error message format.
    /// Placeholders: {field}, {rule}, {value}, {message}
    /// Default: "{field}: {message}"
    /// </summary>
    public string ErrorMessageFormat { get; set; } = "{field}: {message}";
}

```

3. Usage in appsettings.json

Location: samples/ConsoleApp/appsettings.json

```

{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "GSI.IH.Common.Translation": "Debug",

```

```

        "Microsoft": "Warning"
    }
},
"Translation": {
    "EnableCaching": true,
    "CacheDurationMinutes": 10,
    "MaxPayloadSizeKb": 1024,
    "MaxFieldMappings": 500,
    "EnableDetailedLogging": true,
    "TransformationTimeoutSeconds": 30,
    "EnablePerformanceMetrics": true,
    "DefaultValidationProfile": "Strict",
    "EnableAutoTypeConversion": true,
    "ParallelProcessingThreshold": 100
},
"Validation": {
    "Enabled": true,
    "DefaultProfile": "Strict",
    "MaxErrorCount": 50,
    "EnableSchemaValidation": true,
    "EnableMandatoryFieldValidation": true,
    "EnableFormatValidation": true,
    "EnableRangeValidation": true,
    "EnableDateRangeValidation": true,
    "TreatWarningsAsErrors": false,
    "ValidationTimeoutSeconds": 10,
    "ErrorMessageFormat": "{field}: {message}"
}
}

```

4. ServiceCollectionExtensions.cs Update

Add configuration binding to the DI registration:

```

using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.DependencyInjection.Extensions;
using GSI.IH.Common.Translation.Configuration;
using GSI.IH.Common.Translation.Engine;
using GSI.IH.Common.Translation.Parsing;
using GSI.IH.Common.Translation.Transformers;
using GSI.IH.Common.Translation.Validators;
using GSI.IH.Common.Translation.ReferenceData;

namespace GSI.IH.Common.Translation.Extensions;

```

```

public static class ServiceCollectionExtensions
{
    /// <summary>
    /// Adds GSI Translation services with default configuration.
    /// </summary>
    public static IServiceCollection AddGSITranslation(this IServiceCollection se
    {
        return services.AddGSITranslation(options => { });
    }

    /// <summary>
    /// Adds GSI Translation services with configuration.
    /// </summary>
    public static IServiceCollection AddGSITranslation(
        this IServiceCollection services,
        Action<TranslationOptions> configureOptions)
    {
        ArgumentNullException.ThrowIfNull(configureOptions);

        // Register options
        services.Configure(configureOptions);
        services.Configure<ValidationOptions>(options => { });

        // Register core services
        RegisterCoreServices(services);

        return services;
    }

    /// <summary>
    /// Adds GSI Translation services with configuration from IConfiguration.
    /// </summary>
    public static IServiceCollection AddGSITranslation(
        this IServiceCollection services,
        IConfiguration configuration)
    {
        ArgumentNullException.ThrowIfNull(configuration);

        // Bind configuration
        services.Configure<TranslationOptions>(
            configuration.GetSection(TranslationOptions.SectionName));
        services.Configure<ValidationOptions>(
            configuration.GetSection(ValidationOptions.SectionName));

        // Register core services
        RegisterCoreServices(services);
    }
}

```

```

        return services;
    }

private static void RegisterCoreServices(IServiceCollection services)
{
    // Core Services
    services.TryAddSingleton<ITranslationEngine, TranslationEngine>();
    services.TryAddSingleton<IFieldMapper, FieldMapper>();
    services.TryAddSingleton<ITransformerFactory, TransformerFactory>();

    // Parsing Services
    services.TryAddSingleton<IJsonParser, JsonParser>();
    services.TryAddSingleton<IJsonPathResolver, JsonPathResolver>();

    // Validation Services
    services.TryAddSingleton<IValidationOrchestrator, ValidationOrchestrator>

    // Reference Data
    services.TryAddSingleton<IReferenceDataProvider, InMemoryReferenceDataPro

    // Transformers
    services.TryAddEnumerable(new[]
    {
        ServiceDescriptor.Singleton<IFieldTransformer, CopyTransformer>(),
        ServiceDescriptor.Singleton<IFieldTransformer, LookupTransformer>(),
        ServiceDescriptor.Singleton<IFieldTransformer, DateTimeISOTransformer
        ServiceDescriptor.Singleton<IFieldTransformer, ParseLimitStringTransf
        ServiceDescriptor.Singleton<IFieldTransformer, UppercaseTransformer>()
        ServiceDescriptor.Singleton<IFieldTransformer, LowercaseTransformer>()
        ServiceDescriptor.Singleton<IFieldTransformer, ConcatenateTransformer
        ServiceDescriptor.Singleton<IFieldTransformer, ConditionalTransformer
    });

    // Validators
    services.TryAddEnumerable(new[]
    {
        ServiceDescriptor.Singleton<IValidationRule, SchemaValidator>(),
        ServiceDescriptor.Singleton<IValidationRule, MandatoryFieldValidator>
        ServiceDescriptor.Singleton<IValidationRule, RangeValidator>(),
        ServiceDescriptor.Singleton<IValidationRule, FormatValidator>(),
        ServiceDescriptor.Singleton<IValidationRule, DateRangeValidator>()
    });
}

/// <summary>
/// Adds a custom field transformer.

```

```

    /// </summary>
    public static IServiceCollection AddTransformer<TTransformer>(this IServiceCo
        where TTransformer : class, IFieldTransformer
    {
        services.TryAddEnumerable(ServiceDescriptor.Singleton<IFieldTransformer,
            return services;
    }

    /// <summary>
    /// Adds a custom validation rule.
    /// </summary>
    public static IServiceCollection AddValidator<TValidator>(this IServiceCollec
        where TValidator : class, IValidationRule
    {
        services.TryAddEnumerable(ServiceDescriptor.Singleton<IValidationRule, TV
            return services;
    }
}

```

5. Usage Examples

Example 1: Using Configuration from appsettings.json

```

using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using GSI.IH.Common.Translation.Extensions;

var host = Host.CreateDefaultBuilder(args)
    .ConfigureServices((context, services) =>
    {
        // Register GSI Translation with configuration from appsettings.json
        services.AddGSITranslation(context.Configuration);
    })
    .Build();

await host.RunAsync();

```

Example 2: Using Configuration Action

```

using Microsoft.Extensions.DependencyInjection;
using GSI.IH.Common.Translation.Extensions;

```

```

var services = new ServiceCollection();

services.AddGSITranslation(options =>
{
    options.EnableCaching = true;
    options.CacheDurationMinutes = 15;
    options.MaxPayloadSizeKb = 2048;
    options.EnableDetailedLogging = true;
    options.DefaultValidationProfile = "Relaxed";
});

```

Example 3: Using Options Pattern

```

using Microsoft.Extensions.Options;
using GSI.IH.Common.Translation.Configuration;
using GSI.IH.Common.Translation.Engine;

public class MyService
{
    private readonly ITranslationEngine _engine;
    private readonly TranslationOptions _options;
    private readonly ILogger<MyService> _logger;

    public MyService(
        ITranslationEngine engine,
        IOptions<TranslationOptions> options,
        ILogger<MyService> logger)
    {
        _engine = engine;
        _options = options.Value;
        _logger = logger;
    }

    public async Task<string> TranslateAsync(string payload, string definitionJso
    {
        if (_options.EnableDetailedLogging)
        {
            _logger.LogDebug("Starting translation with payload size: {Size}KB",
                payload.Length / 1024);
        }

        // Check payload size
        var payloadSizeKb = payload.Length / 1024;
        if (payloadSizeKb > _options.MaxPayloadSizeKb)

```

```

        {
            throw new InvalidOperationException(
                $"Payload size ({payloadSizeKb}KB) exceeds maximum ({_options.Max
            }

        var definition = JsonSerializer.Deserialize<TranslationDefinition>(defini

        // Check field mapping count
        if (definition!.FieldMappings.Count > _options.MaxFieldMappings)
        {
            throw new InvalidOperationException(
                $"Field mapping count ({definition.FieldMappings.Count}) exceeds
            }

        var result = await _engine.TranslateAsync(
            payload,
            definition,
            validationProfile: _options.DefaultValidationProfile);

        return result.TranslatedPayload;
    }
}

```

6. Configuration Validation

Add DataAnnotations for configuration validation:

```

using System.ComponentModel.DataAnnotations;

namespace GSI.IH.Common.Translation.Configuration;

public sealed class TranslationOptions
{
    public const string SectionName = "Translation";

    public bool EnableCaching { get; set; } = true;

    [Range(1, 1440, ErrorMessage = "Cache duration must be between 1 and 1440 min
    public int CacheDurationMinutes { get; set; } = 10;

    [Range(1, 102400, ErrorMessage = "Max payload size must be between 1KB and 10
    public int MaxPayloadSizeKb { get; set; } = 1024;

    [Range(1, 10000, ErrorMessage = "Max field mappings must be between 1 and 100
    public int MaxFieldMappings { get; set; } = 500;

```

```

public bool EnableDetailedLogging { get; set; } = false;

[Range(1, 300, ErrorMessage = "Timeout must be between 1 and 300 seconds")]
public int TransformationTimeoutSeconds { get; set; } = 30;

public bool EnablePerformanceMetrics { get; set; } = false;

[RegularExpression("^(Strict|Relaxed|None)$",
    ErrorMessage = "Validation profile must be Strict, Relaxed, or None")]
public string DefaultValidationProfile { get; set; } = "Strict";

public bool EnableAutoTypeConversion { get; set; } = true;

[Range(1, 1000, ErrorMessage = "Parallel threshold must be between 1 and 1000
public int ParallelProcessingThreshold { get; set; } = 100;
}

```

7. Complete File List

Now you have **ALL** configuration files:

```

src/GSI.IH.Common.Translation/Configuration/
├─ TranslationOptions.cs          ✓ NOW COMPLETE
└─ ValidationOptions.cs          ✓ NOW COMPLETE

```

Summary

- ✓ **TranslationOptions.cs** - Complete with 10 configuration properties
- ✓ **ValidationOptions.cs** - Complete with 10 configuration properties
- ✓ **ServiceCollectionExtensions.cs** - Updated with configuration binding
- ✓ **appsettings.json** - Example configuration
- ✓ **Usage Examples** - 3 different ways to configure
- ✓ **Configuration Validation** - DataAnnotations for safety

No more missing files! 🎉