MTC_Compression_Test_Import — Test Specifications

This document provides detailed specifications for all unit and component tests implemented in the `MTC_Compression_Test_Import.Tests` project.

Test Framework: xUnit
Assertion Library: FluentAssertions
Mocking Library: Moq
Total Test Methods: 41 (51 test cases including Theory data rows)

---

1. Parser Unit Tests

Test Class: `MTC_Compression_Test_Import.Tests.Parser.CompressionReportParserTests`
System Under Test: `CompressionReportParser.ParseAsync`
Test Count: 19 methods

UT-P-001: ParseAsync rejects null/whitespace path

Attribute	Value
Test Method	`ParseAsync_NullOrWhitespacePath_ThrowsArgumentException`
Test Type	Theory (parameterized)
Requirement	FR-040

Description: Validates that the parser rejects invalid file paths before attempting any file operations.

Test Cases:

Input	Expected Result
`null`	`ArgumentException` with parameter name `filePath`
`""` (empty)	`ArgumentException` with parameter name `filePath`
`" "` (whitespace)	`ArgumentException` with parameter name `filePath`
`"\t"` (tab)	`ArgumentException` with parameter name `filePath`

UT-P-002: ParseAsync rejects missing file

Attribute	Value
Test Method	`ParseAsync_MissingFile_ThrowsFileNotFoundException`
Test Type	Fact
Requirement	FR-040

Description: Validates that the parser throws `FileNotFoundException` when the specified file does not exist on disk.

Setup:

- Generate a random non-existent file path in the temp directory

Expected Result: `FileNotFoundException`

UT-P-003: Missing Compression Tester # in header

Attribute	Value
Test Method	`ParseAsync_MissingTesterIdInHeader_ThrowsInvalidOperationException`
Test Type	Fact
Requirement	FR-041

Description: Validates that the parser requires the `Compression Tester #` identifier in the file header.

Setup:

- Create test file with header lines that do not contain "Compression Tester"
- Include valid data rows

Expected Result: `InvalidOperationException` with message containing "Compression Tester"

UT-P-004: No core data rows parsed

Attribute	Value
Test Method	`ParseAsync_NoDataRows_ThrowsInvalidOperationException`
Test Type	Fact
Requirement	FR-042

Description: Validates that the parser rejects files where no lines can be parsed into the required 6-column format.

Setup:

- Create test file with valid header (includes Tester ID)
- Include lines that do not have 6 columns when split by 4+ spaces

Expected Result: `InvalidOperationException` with message containing "No core data rows"

UT-P-005: No start row found

Attribute	Value
Test Method	`ParseAsync_NoStartRow_ThrowsInvalidOperationException`
Test Type	Fact
Requirement	FR-043

Description: Validates that the parser rejects files where no row matches the start row criteria.

Start Row Criteria: `Col2 == 1` AND `Col5 > 0` AND `Col6 == 0`

Setup:

- Create test file with valid header
- Include 6-column rows that fail start row criteria:
  - Row with `Col2 = 0` (status flag not set)
  - Row with `Col5 = 0` (no pellet number)
  - Row with `Col6 = 1` (end of operation flag set)

Expected Result: `InvalidOperationException` with message containing "No start row"

UT-P-007: Timestamp parse failure

Attribute	Value
Test Method	`ParseAsync_InvalidTimestamp_ThrowsInvalidOperationException`
Test Type	Fact
Requirement	FR-048

Description: Validates that the parser rejects files containing rows with unparseable timestamps.

Setup:

- Create test file with valid header and start row
- Include a row with `Col1 = "not-a-timestamp"` that would otherwise be included in DataPoints

Expected Result: `InvalidOperationException` with message containing "Unable to parse timestamp"

UT-P-010: Incomplete test with pellet reset/repeat

Attribute	Value
Test Method	`ParseAsync_IncompleteTestWithPelletReset_ThrowsInvalidOperationExcep`
Test Type	Fact
Requirement	FR-045

Description: Validates that incomplete tests (no end-of-operation marker) are rejected when pellet numbers reset.

Setup:

- Create test file with valid header and start row
- Include rows with pellet numbers: 1, 2, 3, 1 (reset from 3 to 1)
- No end-of-operation marker row

Expected Result: `InvalidOperationException` with message containing "pellet numbers reset/repeat"

Attribute	Value
Test Method	`ParseAsync_IncompleteTestWithPelletRepeat_ThrowsInvalidOperationExce`
Test Type	Fact
Requirement	FR-045

Description: Validates that incomplete tests are rejected when pellet numbers repeat (non-increasing).

Setup:

- Create test file with valid header and start row
- Include rows with pellet numbers: 1, 2, 2 (repeat)
- No end-of-operation marker row

Expected Result: `InvalidOperationException` with message containing "pellet numbers reset/repeat"

UT-P-011: Incomplete test, last pellet < threshold, strict mode enabled

Attribute	Value
Test Method	`ParseAsync_IncompleteTestBelowThreshold_StrictMode_ThrowsInvalidOper`
Test Type	Fact
Requirement	FR-046

Description: Validates that in strict mode, incomplete tests below the pellet threshold are rejected.

Setup:

- `strictEndOfOperation = true`
- `pelletAcceptanceThreshold = 100.0`
- Create test file with last pellet number = 3 (below threshold)
- No end-of-operation marker

Expected Result: `InvalidOperationException` with message containing "End of operation flag is not 1"

UT-P-012: Incomplete test, last pellet < threshold, lenient mode

Attribute	Value
Test Method	`ParseAsync_IncompleteTestBelowThreshold_LenientMode_Succeeds`
Test Type	Fact
Requirement	FR-046

Description: Validates that in lenient mode, incomplete tests below threshold are accepted with warnings.

Setup:

- `strictEndOfOperation = false`
- `pelletAcceptanceThreshold = 100.0`
- Create test file with 3 measurement rows (last pellet = 3)
- No end-of-operation marker

Expected Result:

- Parse succeeds (no exception)
- `result.EndOfOperationIsOne == false`
- `result.LastPelletNumber == 3.0`
- `result.DataPoints.Count == 3`

Attribute	Value
Test Method	`ParseAsync_IncompleteTestAboveThreshold_LenientMode_Succeeds`
Test Type	Fact
Requirement	FR-046

Description: Validates that incomplete tests above threshold are accepted.

Setup:

- `strictEndOfOperation = false`
- `pelletAcceptanceThreshold = 100.0`
- Create test file with 105 measurement rows (last pellet = 105)
- No end-of-operation marker

Expected Result:

- Parse succeeds
- `result.EndOfOperationIsOne == false`
- `result.LastPelletNumber == 105.0`

UT-P-020: End-of-operation marker row excluded from DataPoints

Attribute	Value
Test Method	`ParseAsync_EndOfOperationMarkerRow_ExcludedFromDataPoints`
Test Type	Fact
Requirement	FR-047

Description: Validates that end-of-operation marker rows are not included in the DataPoints collection.

Marker Row Criteria: `Col2 == 0` AND `Col6 == 1`

Setup:

- Create test file with 3 measurement rows
- Add end-of-operation marker row at the end

Expected Result:

- `result.DataPoints.Count == 3` (marker excluded)
- All DataPoints have `CompressionLbs > 0` (marker row has `Col4 = 0`)
- `result.EndOfOperationIsOne == true`

UT-P-021: Trailing saturated 1000.000 excluded for incomplete tests

Attribute	Value
Test Method	`ParseAsync_IncompleteTestWithTrailingSaturatedValue_ExcludesTrailing`
Test Type	Fact
Requirement	FR-047

Description: Validates that trailing saturated values (1000.000 lbs) are excluded from incomplete tests.

Setup:

- Create test file with 3 normal measurement rows
- Add trailing row with `Col4 = 1000.0`
- No end-of-operation marker

Expected Result:

- `result.DataPoints.Count == 3` (saturated row excluded)
- No DataPoint has `CompressionLbs == 1000.0`
- `result.EndOfOperationIsOne == false`

Attribute	Value
Test Method	`ParseAsync_CompleteTestWithTrailingSaturatedValue_IncludesTrailingRo`
Test Type	Fact
Requirement	FR-047

Description: Validates that saturated values are included when the test is complete.

Setup:

- Create test file with 2 normal measurement rows
- Add row with `Col4 = 1000.0`
- Add end-of-operation marker row

Expected Result:

- `result.DataPoints.Count == 3` (saturated row included)
- DataPoints contains entry with `CompressionLbs == 1000.0`

- `result.EndOfOperationIsOne == true`

UT-P-022: LastPelletNumber/EndOfOperationIsOne derived from last test row

Attribute	Value
Test Method	`ParseAsync_LastPelletAndEndFlag_DerivedFromLastTestRow_NotLastDataPo`
Test Type	Fact
Requirement	FR-047 (note in spec)

Description: Validates that `LastPelletNumber` and `EndOfOperationIsOne` are derived from the last parsed test row, not the last DataPoint.

Setup:

- Create test file with 3 measurement rows (pellet numbers 1, 2, 3)
- Add end-of-operation marker row with `Col5 = 99` (different pellet number)

Expected Result:

- `result.LastPelletNumber == 99.0` (from marker row, not last measurement)
- `result.EndOfOperationIsOne == true`
- `result.DataPoints.Count == 3` (marker excluded)
- `result.DataPoints.Last().PelletNumber == 3` (last measurement pellet)

Additional Coverage Tests

Test Method	Description	E
`ParseAsync_ValidReport_ReturnsCorrectResult`	Validates successful parsing of a complete, valid report	F
`ParseAsync_CancellationRequested_ThrowsOperationCanceledException`	Validates cancellation	

Test Method	Description	E
	token support	
`ParseAsync_RowsWithWrongColumnCount_Ignored`	Validates that malformed rows are silently ignored	F c
`ParseAsync_DataPoints_HaveCorrectValues`	Validates DataPoint property mapping	a

---

2. FileProcessor Component Tests

Test Class: `MTC_Compression_Test_Import.Tests.Processor.FileProcessorTests`
System Under Test: `FileProcessor.RunAsync`
Test Count: 22 methods

Note: All tests use mocked `IFileParser` and `ICompressionRepository` to avoid Oracle database dependencies. Database writes are disabled via constructor parameters.

CT-F-001: Missing options causes exit code 1

Attribute	Value
Test Method	`RunAsync_MissingRequiredOptions_ReturnsExitCode1`
Test Type	Theory (parameterized)
Requirement	FR-010

Description: Validates that missing or empty required directory options cause the processor to return exit code 1.

Test Cases:
SourceDirectory	ArchiveDirectory	ErrorDirectory	Expected

`null`	"archive"	"error"	Exit code 1
`""`	"archive"	"error"	Exit code 1
`" "`	"archive"	"error"	Exit code 1
"source"	`null`	"error"	Exit code 1
"source"	`""`	"error"	Exit code 1
"source"	`" "`	"error"	Exit code 1
"source"	"archive"	`null`	Exit code 1
"source"	"archive"	`""`	Exit code 1
"source"	"archive"	`" "`	Exit code 1

CT-F-002: Missing source directory causes exit code 1

Attribute	Value
Test Method	`RunAsync_SourceDirectoryDoesNotExist_ReturnsExitCode1`
Test Type	Fact
Requirement	FR-020

Description: Validates that a non-existent source directory causes exit code 1.

Setup:

- Configure `SourceDirectory` to point to a non-existent path
- Create archive and error directories

Expected Result: Exit code 1

CT-F-003: Missing archive/error directories are created

Attribute	Value
Test Method	`RunAsync_ArchiveDirectoryDoesNotExist_CreatesDirectory`
Test Type	Fact
Requirement	FR-021

Description: Validates that a missing archive directory is automatically created.

Setup:

- Create source directory
- Do not create archive directory
- Create error directory

Expected Result: Archive directory exists after `RunAsync` completes

---

Attribute	Value
Test Method	`RunAsync_ErrorDirectoryDoesNotExist_CreatesDirectory`
Test Type	Fact
Requirement	FR-022

Description: Validates that a missing error directory is automatically created.

Setup:

- Create source and archive directories
- Do not create error directory

Expected Result: Error directory exists after `RunAsync` completes

Attribute	Value
Test Method	`RunAsync_BothArchiveAndErrorDirectoriesDoNotExist_CreatesBoth`
Test Type	Fact
Requirement	FR-021, FR-022

Description: Validates that both archive and error directories are created when missing.

Setup:

- Create source directory only

Expected Result: Both archive and error directories exist after `RunAsync` completes

---

CT-F-004: Duplicate filename present in archive

Attribute	Value
Test Method	`RunAsync_DuplicateFilenameInArchive_MovesFileToErrorDirectory`
Test Type	Fact
Requirement	FR-024

Description: Validates that files with duplicate names in the archive are moved to the error directory without parsing.

Setup:

- Create `test_report.txt` in source directory with content "source content"
- Create `test_report.txt` in archive directory with content "archive content"

Expected Result:

- Parser is never called
- Source file no longer exists in source directory
- File exists in error directory
- Archive file unchanged (content = "archive content")

CT-F-010: Retry on IOException

Attribute	Value
Test Method	`RunAsync_IOExceptionOnFirstAttempt_RetriesAndSucceeds`
Test Type	Fact
Requirement	FR-030, FR-031

Description: Validates that `IOException` triggers retry and processing succeeds on second attempt.

Setup:

- Mock parser throws `IOException` on first call
- Mock parser returns successful result on second call

Expected Result:

- Exit code 0
- Parser called exactly 2 times
- File moved to archive directory

CT-F-011: Retry on UnauthorizedAccessException

Attribute	Value
Test Method	`RunAsync_UnauthorizedAccessExceptionOnFirstAttempt_RetriesAndSucceed`
Test Type	Fact
Requirement	FR-030, FR-031

Description: Validates that `UnauthorizedAccessException` triggers retry and processing succeeds on second attempt.

Setup:

- Mock parser throws `UnauthorizedAccessException` on first call
- Mock parser returns successful result on second call

Expected Result:

- Exit code 0
- Parser called exactly 2 times
- File moved to archive directory

CT-F-010b: Retry exhaustion moves file to error

Attribute	Value
Test Method	`RunAsync_IOExceptionOnAllAttempts_MovesFileToError`
Test Type	Fact
Requirement	FR-030, FR-032

Description: Validates that after all retry attempts are exhausted, the file is moved to error.

Setup:

- Mock parser throws `IOException` on every call

Expected Result:

- Exit code 0 (processing continues with other files)
- Parser called exactly 3 times (max retries)
- File moved to error directory

CT-F-012: Non-retry exception moves file to error

Attribute	Value
Test Method	`RunAsync_InvalidOperationException_MovesFileToErrorWithoutRetry`
Test Type	Fact
Requirement	FR-032

Description: Validates that `InvalidOperationException` (non-retryable) moves file to error immediately.

Setup:

- Mock parser throws `InvalidOperationException`

Expected Result:

- Exit code 0 (processing continues)
- Parser called exactly 1 time (no retry)
- File moved to error directory

Attribute	Value
Test Method	`RunAsync_ArgumentException_MovesFileToErrorWithoutRetry`
Test Type	Fact
Requirement	FR-032

Description: Validates that `ArgumentException` (non-retryable) moves file to error immediately.

Setup:

- Mock parser throws `ArgumentException`

Expected Result:

- Parser called exactly 1 time (no retry)
- File moved to error directory

Additional Coverage Tests

Test Method	Requirement	Description
`RunAsync_NoFilesInSourceDirectory_ReturnsExitCode0`	FR-023	Empty source directory
`RunAsync_SuccessfulProcessing_MovesFileToArchive`	—	Successful file processing
`RunAsync_MultipleFiles_ProcessesAll`	—	Multiple files in source
`RunAsync_CancellationRequested_ReturnsExitCode1`	FR-033	Cancellation token signaled
`RunAsync_FilePatternFilter_OnlyProcessesMatchingFiles`	FR-023	Mixed file types with `*.txt` filter

CT-F-020: Error categorization and logging

Attribute	Value
Test Method	`RunAsync_ParsingError_CategorizedCorrectly`
Test Type	Fact
Requirement	—

Description: Validates that parsing errors (e.g., "No start row") are categorized correctly and file is moved to error directory.

Attribute	Value
Test Method	`RunAsync_MissingHeaderError_CategorizedCorrectly`
Test Type	Fact
Requirement	—

Description: Validates that missing header errors are categorized correctly.

Attribute	Value
Test Method	`RunAsync_MultipleFailures_AllRecordedInSummary`
Test Type	Fact
Requirement	—

Description: Validates that multiple file failures are all recorded and processing continues.

Setup:

- Create 3 files: 2 that will fail parsing, 1 that succeeds

Expected Result:

- Exit code 0
- 2 files in error directory

- 1 file in archive directory

Attribute	Value
Test Method	`RunAsync_DuplicateFile_RecordedAsError`
Test Type	Fact
Requirement	FR-024

Description: Validates that duplicate files are recorded as errors with "Duplicate" category.

Attribute	Value
Test Method	`RunAsync_PelletSequenceError_CategorizedAsValidationError`
Test Type	Fact
Requirement	—

Description: Validates that pellet sequence errors are categorized as validation errors.

---

3. Test Helpers

TestFileBuilder

Location: `MTC_Compression_Test_Import.Tests.Helpers.TestFileBuilder`

A fluent builder class for programmatically generating test report content.

Methods:
Method	Description
`WithTesterId(int id)`	Adds header line with `Compression Tester #{id}`
`WithStandardHeader()`	Adds standard column headers
`WithHeaderLine(string line)`	Adds arbitrary header line
`WithDataRow(string timestamp, double col2, double col3, double col4, double col5, double col6)`	Adds formatted 6-column data row
`WithRawLine(string line)`	Adds arbitrary line

| `Build()` | Returns the complete file content string |
| `CreateValidReport(int testerId, double testNumber, int measurementCount)` | Static factory for valid complete reports |

---

4. Test Execution

Running Tests

```
# Run all tests
dotnet test MTC_Compression_Test_Import.Tests

# Run with verbose output
dotnet test MTC_Compression_Test_Import.Tests --logger
"console;verbosity=detailed"

# Run specific test class
dotnet test MTC_Compression_Test_Import.Tests --filter
"FullyQualifiedName~CompressionReportParserTests"

# Run specific test
dotnet test MTC_Compression_Test_Import.Tests --filter
"FullyQualifiedName~ParseAsync_NullOrWhitespacePath"
```

Test Coverage Summary

Category	Test Methods	Test Cases	Requirements Covered
Parser Unit Tests	19	25	FR-040 through FR-048
FileProcessor Component Tests	22	26	FR-010, FR-020 through FR-033
Total	**41**	**51**	—

---

5. Not Implemented (Deferred)

The following test categories from the validation spec are not implemented per project requirements:

Integration Tests (IT-D-*)

- IT-D-001: Insert writes expected header/detail rows
- IT-D-002: Preview mode does not write
- IT-D-003: Stats calculation and rounding
- IT-D-004: Post-insert failure side effect

Reason: Requires Oracle test database connection

Acceptance Tests (AT-*)

- AT-001: Compare stats against production reference

Reason: Requires golden data comparison with production system