# **SSIS Wireframe Failed File Debugging Guide**

## **Introduction**

When working with SSIS (SQL Server Integration Services) WireFrame extracts, it is common to encounter failed jobs due to environment mismatches, incorrect parameters, configuration issues, or data-related problems. To ensure smooth recovery and minimal downtime, it is critical to follow a structured debugging process.

This document outlines a step-by-step approach for identifying failures, validating configurations, testing extracts in a controlled environment, and ensuring fixes are correctly applied across Test and Production environments. The goal is to provide clear guidance for developers and support teams on how to efficiently debug and resolve failed extracts in SSIS WireFrame.

## **Steps to Follow**

## **1. Identify Failed Extracts from Log Table**

* Navigate to the log table **RawReporting.EtlErrorLog**, which records all extract activities (successful or failed).
* Use a query to filter out errors for the date in question:

sql

SELECT \*

FROM RawReporting.EtlErrorLog

WHERE CAST(Instant AS DATE) = {desired\_date}

AND Severity = 'Error';

* From the result set:
  + Note the Definition\_ID of failed extracts.
  + Review the ErrorDescription column for details on why each job failed.

## **2. Review Failure Details**

* Carefully analyze the ErrorDescription to classify the issue:
  + Missing or incorrect parameters
  + Incorrect file path or destination
  + Data-related errors (null values, truncations, schema mismatches)
  + Server configuration or connectivity issues, etc.

## **3. Validate Time Dependency of Extracts**

* Before attempting to rerun, check if the extract is time-dependent:
  + Query the **Parameter\_Config\_Tbl** table for the failed Definition\_IDs.
  + Look for extracts that accept Date Parameters.
* For time-dependent extracts (daily/monthly runs):
  + Daily extracts: Failure reruns may produce different results unless parameters are fixed to the original date.
  + Monthly extracts: Only reruns from different months will differ, so rerunning with the same parameters should match expected results.
* Note down the parameter values required to replicate the exact run that was intended before failure.

## **4. Validate Test Environment Configurations**

* Move to the Test Server for initial debugging.
* In the SSISConfigTable:
  + Ensure that Definition\_ID entries match those on Production, except for Destination\_Location (this should point to test-specific folders).
* In the ParameterConfig Table:
  + Confirm that parameter values are correctly defined.
  + Update or overwrite values as needed to match the failed run’s configuration.

## **5. Make Corrections and Test the Extract**

* Based on the EtlErrorLog message:
  + Apply necessary changes in Test Server configuration.
  + Examples:
    - Correct parameter values.
    - Modify column mappings or data type expectations.
* Run the extract manually using the WireFrame Adhoc Run Job for the Definition\_ID.

## **6. Validate Results with Developer**

* If the test run succeeds, check the SSISConfigTable for the assigned Developer responsible for the extract.
* Notify the developer via email, attaching or pointing to the generated test file.
* Request the developer to validate whether the output matches business requirements.

## **7. Apply Fixes in Production and Rerun**

* Once validated by the developer:
  + Apply the same corrections in the Production Server’s configuration tables.
  + Run the extract from production with the correct parameters.
* Confirm that the extract completes without errors and generates the expected outputs.

## **8. Revert Temporary Parameter Changes**

* If any changes were applied to Date Parameters or other temporary configurations:
  + Revert them back to default values in both Test and Production servers.
* This ensures future runs are clean and not dependent on debugging modifications.

## **Additional Best Practices**

* Always document changes: Note down the Definition\_ID, error message, fixes applied, and confirmation status. This helps build a knowledge base for recurring issues.
* Version control your configurations: If possible, export or track configuration table changes in version control systems.
* Proactive monitoring: Set up automated alerts from RawReporting.EtlErrorLog so that failures are flagged quickly without waiting for manual checks.
* Minimize production disruptions: Always test thoroughly in the test environment before applying changes to production.