# Step-by-Step Logic of Export Status Update and ResolvedInErrorQueue Handling

This document explains in sequential order the SQL script logic provided, covering the creation of temporary tables, updates from ErrorResponse, ErrorQueue handling, and finally how ExportStatus is updated with 'ResolvedInErrorQueue'.

## 1. Insert Into #Results

The script begins by inserting distinct authorization records into the temporary table #Results from the Exports.Canonical.[Authorization] table and related joins with AuthorizationDestination and ExportProcess.

## 2. Insert Into #AuthorizationDestination

A temporary table #AuthorizationDestination is created by joining #Results with Authorization, AuthorizationDestination, ExportProcess, and AuthorizationDestinationStatus. This table provides information about the export destination status for each authorization.

## 3. Update #Results with Destination Data

The #Results table is updated with DestinationDate and DestinationStatus values from #AuthorizationDestination. This links the authorization with its most recent destination processing information.

## 4. Insert Most Recent Success into #Success

The script selects the most recent success response for each authorization from Exports.EDI.SuccessResponse and inserts it into #Success. Duplicate records are removed, keeping only the latest InsertDate per AuthorizationKey. Then #Results is updated with this SuccessDate.

## 5. Insert Most Recent Error into #Error

Errors are retrieved from Exports.EDI.ErrorResponse and inserted into #Error. Only the most recent error (based on InsertDate) is retained. Then #Results is updated with ErrorDate, ErrorCode, and ErrorDescription.

## 6. ExportStatus Calculation in #Results

Based on DestinationDate, DestinationStatus, SuccessDate, and ErrorDate, the ExportStatus is calculated:  
- Exported → PendingResponse or Success (based on subsequent dates)  
- Pending → PendingExport  
- SelfDiscoveryError → Error  
- ExportNotRequired → NotExported  
- If SuccessDate is latest → Success  
- If ErrorDate is latest → Error  
- Otherwise → NotExported

## 7. Special Business Rules

Two special cases are handled:  
- CT errors (Cigna past lifespan) are considered 'Success'.  
- EVIS0019 errors are considered 'PendingResponse'.  
- SelfDiscoveryError-related cases remain 'Error'.

## 8. Create #TempErrorQueue

A temporary table #TempErrorQueue is created from Exports.Canonical.ErrorQueue joined with Error, ErrorCategory, ErrorType, ExportProcess, ErrorStatus, and Authorization. This captures detailed error queue data for authorizations.

## 9. Cleanup #TempErrorQueue

Records are deleted from #TempErrorQueue where:  
- ErrorCode does not match with #DailyAuthExportStatus.ErrorCode.  
- ErrorDate from #DailyAuthExportStatus is older than #TempErrorQueue.LastModifiedDate by more than 60 minutes.

## 10. Create #DailyAuthExportStatus2

A new temp table #DailyAuthExportStatus2 is created by joining #DailyAuthExportStatus with #TempErrorQueue. Duplicate rows are resolved using OUTER APPLY with MAX(ErrorQueueId).

## 11. Update to ResolvedInErrorQueue

Finally, ExportStatus is updated to 'ResolvedInErrorQueue' in two places:  
- In #DailyAuthExportStatus: If an AuthorizationID exists in #DailyAuthExportStatus2 with ErrorQueue\_Status = 'Resolved'.  
- In #Results: Similarly, if AuthorizationID exists with ErrorQueue\_Status = 'Resolved', then ExportStatus is updated.

## 12. ErrorResponse vs ErrorQueue Interaction

a) If ErrorResponse has no records for an AuthorizationID → #Error is empty, so #Results has no error fields. ExportStatus depends only on Destination and Success.  
  
b) If ErrorResponse has records → #Error is populated with latest InsertDate/ErrorCode/ErrorDescription, which updates #Results. This impacts ExportStatus calculation.  
  
c) ErrorQueue provides high-level error tracking and resolution status. ErrorResponse provides detailed error data from downstream. Both are linked via AuthorizationKey/AuthorizationID. When an error is marked 'Resolved' in ErrorQueue, ExportStatus is updated to 'ResolvedInErrorQueue'.

## 13. Summary

The SQL script ensures a structured flow: Results → Destination → Success/Error → Status Calculation → ErrorQueue Resolution. By combining ErrorResponse details and ErrorQueue resolution statuses, the logic ensures that the final ExportStatus reflects both system errors and business handling of resolved cases.