ERP-1691 Icertis - Integration Retry Job

Status	Approved
Functional Specification Owner	RAMTEK-ext, Vaibhav
Stakeholders	RAMTEK-ext, Vaibhav MARATHE-ext, Aradhana
Jira Request ID	ERP-1324 - Jira project doesn't exist or you don't have permission to view it.
Jira Development (Build) ID	ERP-1691 - Jira project doesn't exist or you don't have permission to view it.

High-Level Specification

Parameter	Value
Application System (Delivery Tool)	Icertis
Business Process Reference (L4)	NA

Summary

Syensqo's integration between Icertis and other systems is critical for smooth business operations. However, the current Icertis setup has significant limitations: failed integrations are hard to track, there's no easy way to retry them after Icertis retries in quick succession, and resolving issues often requires IT support or raising tickets. This leads to delays, lack of transparency, and potential business disruption.

To address these challenges, a new reconciliation mechanism is being introduced. This solution will automatically log and retry failed messages, notify stakeholders of persistent issues, and provide real-time visibility for authorized users. The result is improved reliability, faster issue resolution without relying on technical intervention for every problem.

Functional Overview

The reconciliation mechanism ensures reliability and traceability for custom integrations (e.g., Icertis to CPI) by logging failed messages, managing scheduled retries, tracking retry attempts, and notifying stakeholders of persistent failures.

What Are the Drawbacks of the Current Icertis Integration Process?

1. **Limited Retry Capabilities** : Icertis does not provide a flexible, user-controlled way to automatically retry failed integrations. If a message fails, there is no built-in mechanism for custom or scheduled retries.
2. **Lack of Visibility :** When an integration fails, business or functional users cannot easily see the details or status of failed messages. There is no user-friendly dashboard or report for monitoring these failures in real time.
3. **Manual and Reactive Issue Resolution :** To investigate a failed integration, users often need to raise a support ticket with Icertis or rely on IT teams to check technical logs. This process is slow, reactive, and can delay business operations.

Objectives

- **Reliability:** Ensure that all failed integration messages are captured, retried, and escalated as needed to minimize data loss and integration gaps.
- **Traceability:** Maintain a comprehensive audit trail for each integration attempt, including message content, timestamps, error details, and a record of retries.
- **Timely Escalation:** Automatically notify relevant stakeholders when persistent failures occur, enabling prompt manual intervention.
- **User Transparency:** Provide authorized users with real-time visibility into the status of integration messages for proactive monitoring and troubleshooting.
- **Compliance:** Adhere to Syensqo's IT security, data privacy, and regulatory requirements throughout the process.
- **Reporting:** Enable robust reporting and analytics on integration performance, failure rates, and resolution times to support continuous improvement

Assumptions

- Source systems (e.g., Icertis) are responsible for initial message transmission and may perform immediate retries, but scheduled reconciliation is handled separately.
- The Reconciliation and ReconciliationRetry table via Icertis masterdata is the single source of truth for failed integration attempts.
- Scheduled jobs (e.g., every 4 hours) are configured and maintained by Icertis Admin.
- Users have appropriate access to view failed message statuses via the Icertis UI.
- All integrations adhere to Syensqo's security and compliance policies.

Dependencies

- Scheduled batch jobs must run after source systems have completed their integration attempts.
- The Icertis email notification system must be operational for timely stakeholder alerts.

Special Requirements

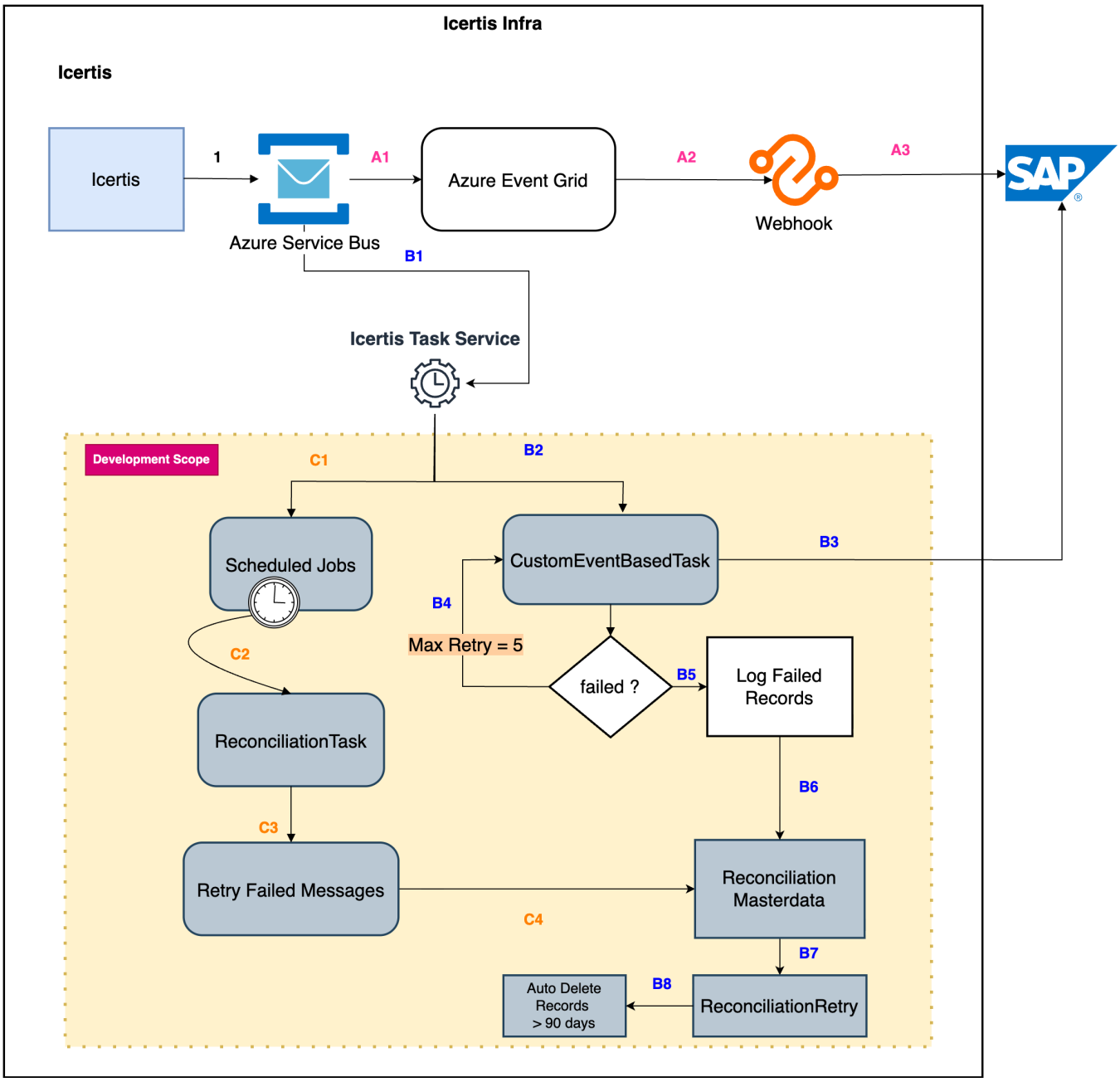
- Integration with third-party systems (Icertis, CPI) requires secure authentication (e.g., OAuth, certificates).
- Audit trails must be maintained for all message processing and status changes.
- The solution must use ETags for optimistic concurrency control, as all API calls internally in Icertis uses ETags for CRUD operations.
- All error messages and notifications must be clear, actionable, and compliant with internal communication standards.

Scope

This development covers the design, implementation, and monitoring of a robust reconciliation mechanism for custom integrations, such as the Icertis to CPI interface. The solution ensures reliable, transparent, and compliant message processing across both internal and external systems. The key aspects include:

- **Centralized Failure Logging:**
  All failed integration messages are automatically logged in dedicated reconciliation tables, capturing message content, timestamps, error details, and retry counts. This serves as the single source of truth for all failed attempts.
- **Automated, Scheduled Retry:**
  A scheduled job runs every 4 hours, retrieving failed messages from the reconciliation tables and attempting to resend them to the target endpoint. Each retry is tracked, with the retry count and status updated after every attempt.
- **Comprehensive Audit Trail:**
  Every retry attempt creates a new entry in the ReconciliationRetry table, maintaining a detailed history of all actions taken for each message.
- **Timely Escalation and Notification:**
  If a message fails more than five times, an automated notification is sent to relevant stakeholders, ensuring timely awareness and enabling prompt manual intervention.
- **User Visibility and Transparency:**
  Authorized users can view the status of failed integrations directly via the Icertis UI, promoting proactive monitoring and troubleshooting.
- **Automated Cleanup:**
  The mechanism includes automatic deletion of reconciliation records older than 90 days, ensuring efficient data management and compliance with retention policies.
- **Compliance and Reporting:**
  The solution adheres to Syensqo's IT security, data privacy, and audit requirements. It also supports reporting and analytics needs as defined in related Jira requests, enabling continuous improvement.

Component Diagram



Sequence Breakdown

1. **Icertis Initiation (1):** The process begins with "Icertis," which sends messages to the "Azure Service Bus." This is the starting point for events or data that need to be processed.
2. **Azure Service Bus to Azure Event Grid (A1):** Messages from the Azure Service Bus are then directed to the "Azure Event Grid."
3. **Azure Event Grid to Webhook (A2):** The Azure Event Grid routes these events to a Icertis Webhook. This implies that the events are being pushed to an external service or application via an HTTP callback.
4. **Webhook to SAP (A3):** The Webhook sends the information to "SAP" for sending updates or new data.

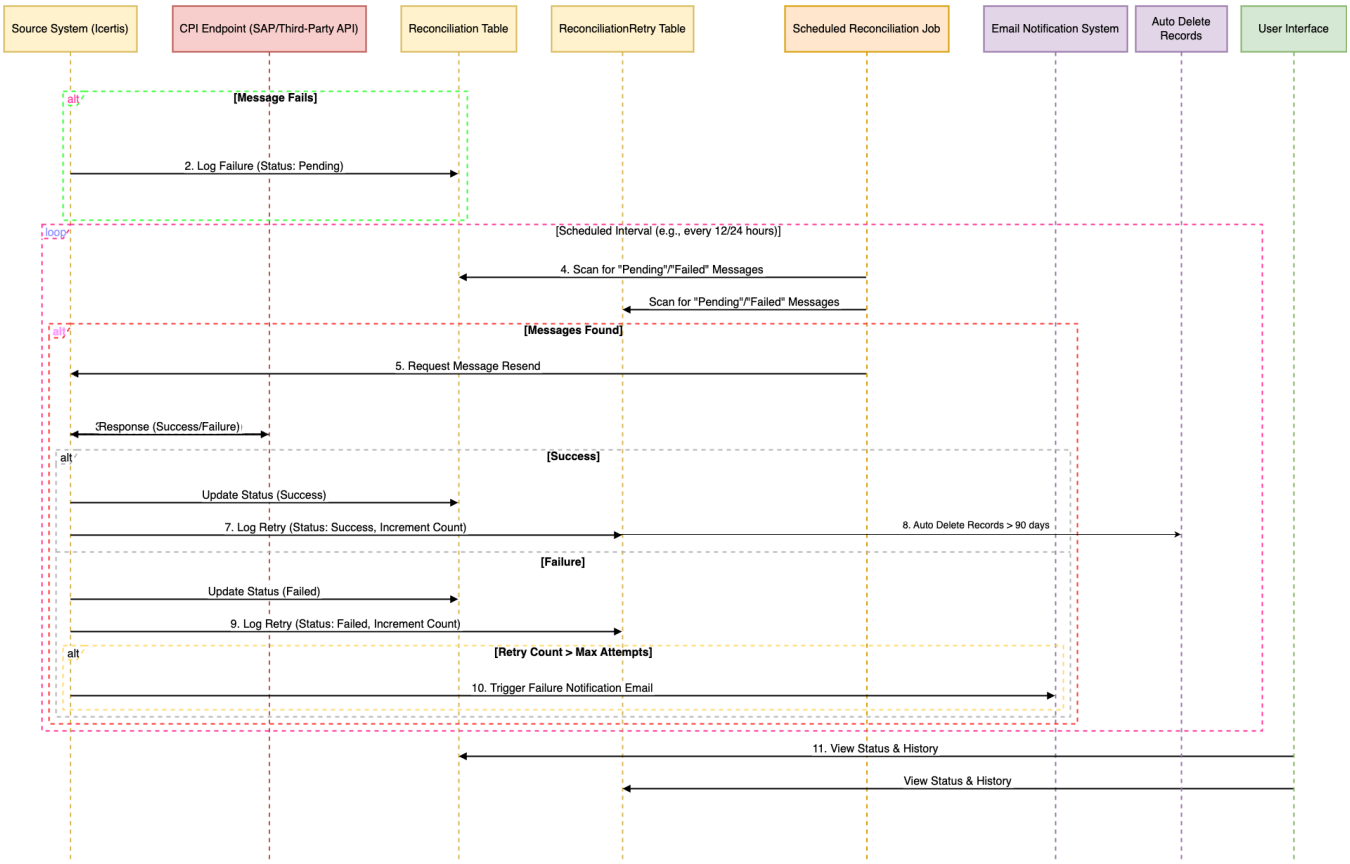
Icertis Task Service and Development Scope :

This section seems to describe a more detailed, internal process within Icertis, particularly focusing on handling events and ensuring data consistency. The "Icertis Task Service" acts as an orchestrator for the processes within the "Development Scope."

1. **Icertis Task Service Orchestration (B1 & B2):** The "Icertis Task Service" is windows service that triggers or manages icertis event based and scheduled tasks within the "Development Scope." It triggers the "Scheduled Jobs" (B2) and e.g. "CustomEventBasedTask" which sends messages to SAP sytem(B3).
2. **CustomEventBasedTask (B3):** The CustomEventBasedTask which invokes the 3rd Party API for integration.
3. **Max Retry = 5 (B4):** If the task fails, there's a retry mechanism with a Max Retry = 5 within Icertis via hangfire. The CustomEventBasedTask will attempt to re-execute up to 5 times if it encounters failures.

4. **Conditional Logic (B5):** If the API call fail, Log the fail record in the Reconciliation table in B6. Regardless of retries, if a task fails, the "Log Failed Records" process is activated to record details of the failure.
5. **Reconciliation Table (B6):** The logged failed records feeds into Reconciliation table.
6. **Auto Delete (B8) : Auto delete the reconciliation and reconciliation retry records older than 90 days.**
7. **ReconciliationTask (C2):** These scheduled jobs trigger a "ReconciliationTask" to process the failed records for the integrations and add /update the ReconciliationRetry table.
8. **Retry Failed Messages (C3 & C4):** If the ReconciliationTask loads all the failed records from the Reconciliation table and reprocess them to 3rd Party API

Sequence Diagram



Step-by-Step Flow of the Reconciliation Mechanism

1. **Integration Attempt**
    - Icertis (or another system) sends a message to the CPI endpoint (e.g., SAP or a third-party API).
2. **Failure Detection**
    - If the message fails (e.g., due to a network error or endpoint issue), the failure is immediately logged in the **Reconciliation** table with all relevant details.
3. **Immediate Retry by Source System (Optional)**
    - The source system (e.g., Icertis) may attempt a quick, immediate retry.
    - **Note:** This immediate retry is separate from the scheduled reconciliation process.
4. **Scheduled Reconciliation Job**
    - At scheduled intervals (e.g., 4 hours), a reconciliation job scans the **Reconciliation** and **ReconciliationRetry** tables for messages with status “Pending” or “Failed.”
    - The job attempts to resend these failed messages to the endpoint.
5. **Logging Retry Attempts**
    - After each retry attempt, a new entry is created in the **ReconciliationRetry** table:
        - The retry count is incremented.
        - The status is updated (e.g., “Retried,” “Success,” or “Failed”).
        - The timestamp of the attempt is recorded.
6. **Failure Escalation**

- If a message's retry count exceeds the maximum allowed attempts (e.g., 5), the system automatically triggers a failure notification email to stakeholders.

7. **User Monitoring**
    - Users can access both the **Reconciliation** and **ReconciliationRetry** tables (e.g., via the Icertis UI) to view the status and history of all failed or retried messages.
8. **Auto Delete aged entries :** Auto delete the reconciliation and reconciliation retry records older than 90 days

Changes Required

This section covers the changes required in the iCertis system to implement the reconciliation job

- **New Masterdata Contract type creation**
    - **Create Masterdata Contract Type:**
        - Reconciliation Masterdata: Stores all failed integration attempts with detailed metadata (message, timestamps, error, retry count, status, etc.).
        - ReconciliationRetry Masterdata: Logs each retry attempt, including attempt number, timestamps, error messages, and success/failure status.
    - **Auto-Deletion Logic:**
        - Implement scheduled cleanup to delete records older than 90 days from both masterdata sets.
        - Ensure audit logging of deletion actions
- **Integration Logic**
    - **Failure Logging:**
        - On integration failure (e.g., Icertis to CPI), log the failure in the Reconciliation masterdata with all relevant details.
        - The error code of the integration to be added in the a Reconciliation Retry Masterdata
    - **Retry Mechanism:**
        - Scheduled job (e.g., every 4 hours) scans for failed messages and attempts resending.
        - Each retry attempt is logged in the ReconciliationRetry masterdata.
        - Retry count is incremented and status updated after each attempt.
        - Max retry limit (e.g., 5) is enforced.
- **Notification & Escalation**
    - **Automated Email Alerts:**
        - If a message fails more than the allowed number of retries, send an automated notification to Icertis Admin users.
        - The Email Id DL needs to be finalized with Sascha
    - **Master Data Access to Icertis Admin Role**
        - Provide view only access to Icertis Admin to view Reconciliation and ReconciliationRetry Masterdata

Configuration

To ensure robust, reliable, and transparent integration between Icertis and downstream systems (such as CPI), it is essential to track and manage all integration attempts—especially failures. The Reconciliation & ReconciliationRetry table is designed to serve as the central repository for logging every failed integration message, capturing key details such as message content, timestamps, error information, retry attempts, and processing status. These table enables:

- **Automated Retry and Escalation:** By recording each failure and retry, the system can automatically reprocess failed messages and escalate persistent issues to stakeholders.
- **Traceability and Auditability:** Every integration attempt is logged with comprehensive details, supporting audit requirements and root cause analysis.
- **User Visibility:** Authorized users can monitor the status of integration messages in real time, enabling proactive intervention and troubleshooting.
- **Reporting and Compliance:** The structured data supports analytics, reporting, and compliance with Syensqo's IT and regulatory standards.

Reconciliation Table

Column Name	Data Type	Nullable?	Description/Notes
SysId	UniqueIdentifier	No (not null)	
Custom TaskId	int	No (not null)	Task id of the integrations
IntegrationType	nvarchar(150)	No (not null)	String holding the integration name.

EntityName	nvarchar (400)	Yes	Icertis Entity Name.
IsSuccess	bit	Yes (null)	A Boolean flag (0 or 1) indicating if the operation succeeded.
ExternalIdentifier	nvarchar(400)	Yes (null)	An Id from an external system. (e.g. Incoming SAP Event/Message Id)
ICMIdentifier	nvarchar(400)	Yes (null)	Id from Icertis system like SysId.
ETag	datetime	Yes (null)	A timestamp or version indicator used for optimistic concurrency control. (default column of Icertis)
IsActive	int	No (not null)	An integer flag (likely 0 or 1) indicating whether the log entry is active. Inactive records are not shown in the Icertis table view UI by default.

ReconciliationRetry Table

Column Name	Data Type	Nullable	Description
SysId	UniqueIdentifier	No (not null)	
ReconciliationID	bigint	No	Foreign Key. References the SysID in your reconciliation table.
RetryAttemptNumber	int	No	The specific iteration (e.g., 1, 2...5).
Reconciliation Message	nvarchar(max)	Yes	Captures the specific message/error that occurred during this retry.
Payload	nvarchar(max)	No	Captures the payload which was sent and retried in next attempt
Error Code	nvarchar(20)	Yes	Captures the error code if error returned from endpoint
StartedAt	datetime	Yes	When this specific attempt began.
CompletedAt	datetime	Yes	When this specific attempt finished.
IsSuccess	bit	Yes	Did this specific attempt succeed?

Batch Job Configuration & Scheduling

Field	Value
Name	SywayReconciliationScheduledTask
POD	S2P
Scheduling Tool	Icertis
Job Type	Batch
Frequency Scheduled	Every 4 hrs
Technical Owner	Icertis Support

Sample Data logging

- Reconciliation Masterdata

Column Name	Sample Value
SysId	737d71-e89d-4581-bf29-bd30ebd29525
TaskId	1001
IntegrationType	SAPContractSync
EntityName	Contract
IsSuccess	1
ExternalIdentifier	SAP-EVENT-20240612-001
ICMIdentifier	ICM-SYS-987654
ETag	2024-06-12 10:15:30
IsActive	1

- ReconciliationRetry Masterdata

SysId	RetryID	ReconciliationID	RetryAttemptNumber	Message	ErrorCode	StatusCode	StartedAt	CompletedAt	IsSuccess
50e8400-e29b-41d4-a716-446655440000	1	737d71-e89d-4581-bf29-bd30ebd29525	1	Timeout error: SAP system did not respond.	504	TIMEOUT	2024-06-12 10:20:00	2024-06-12 10:21:00	0
123e4567-e89b-12d3-a456-426614174000	2	737d71-e89d-4581-bf29-bd30ebd29525	2	Network issue: Connection reset by peer.	502	NETWORK_FAILURE	2024-06-12 10:22:00	2024-06-12 10:23:00	0
550e8400-e29b-41d4-a716-446655440000	3	737d71-e89d-4581-bf29-bd30ebd29525	3	Success: Data reconciled successfully.	200	SUCCESS	2024-06-12 10:24:00	2024-06-12 10:24:30	1

Proposed Technology to Use

- Icertis custom event job & Scheduled Job

Processing Logic

- **When to Run**
    - Reconciliation jobs are scheduled to run at regular intervals (e.g., every 4 hours).
    - These jobs are triggered automatically by the system to process failed or pending integration attempts.
- **How Reconciliation Table Entries Are Added**
    - Users can monitor the status of integration attempts and retries through the Icertis UI or other master data interfaces.
    - Both the Reconciliation and ReconciliationRetry tables are accessible for review, allowing users to:
        - Track the progress and outcome of each integration attempt.
        - View detailed error messages and retry history.
        - Identify records that require manual intervention or escalation.
- **Scheduled Retry**
    - The scheduled reconciliation job scans the **Reconciliation** table for entries with a status of “Pending” or “Failed.”
    - For each failed entry:
        - The job attempts to resend the message to the target endpoint.
        - Each retry attempt is logged in the **ReconciliationRetry** table, incrementing the retry count and updating the status and timestamps.
        - If the retry count exceeds the maximum allowed attempts (e.g., 5), an automated failure notification is sent to stakeholders.
- **User Monitoring**
    - Users can monitor the status of integration attempts and retries through the Icertis UI or other master data interfaces.
    - Both the **Reconciliation** and **ReconciliationRetry** tables are accessible for review, allowing users to:
        - Track the progress and outcome of each integration attempt.
        - View detailed error messages and retry history.
        - Identify records that require manual intervention or escalation.
- **During Scheduled Retry Jobs:**
    - Each time the scheduled reconciliation job runs (e.g., 4 hours), it also checks for and deletes entries older than 90 days from both tables.

How to Test

Test Step ID	Test Condition/Action	Expected Result
1	Simulate a failed integration (e.g., send invalid data or disable the endpoint)	Entry created in the IntegrationReconciliation table with status "Failed" and TryCount = 1
2	Allow the scheduled reconciliation job to run	System attempts resend; TryCount increments; status updates to "Success" if successful, else remains "Failed"/"Retried"
3	Force repeated failures until TryCount > threshold (e.g., 3)	Automated notification sent to stakeholders; record flagged for manual intervention
4	Log in as an authorized user and view the monitoring interface	All integration attempts (failed, retried, and successful) are visible with accurate details
5	Attempt unauthorized access or modification	Access is denied; security controls are enforced
6	Test with corrupted or incomplete data	System handles exceptions gracefully; error is logged appropriately

7	Reset test data and repeat tests	Process is repeatable and robust; system continues to function as expected

Language Requirements

- All labels, logs, and error messages will be in English.

Development Details

Package

Package Name	Parent Package

Enhancement Implementation

Enhancement Type	Standard Definition Name	Custom Implementation Name	Design Rationale Reference

Other Development Objects

Object Type	Object Name	Purpose/High Level Logic	Design Rationale Reference

Workflow history

This view shows the 5 most recent entries. The complete workflow log is available from the 'Document Activity' menu item.

Mar 31, 2026	Actor	Type	Activity	Version
Approved	WENNINGER-ext, Sascha	State	changed state to Approved at 8:29 am	v64
Revision under Review	WENNINGER-ext, Sascha	State	gave *Minor change* approval at 8:29 am *minor change to field label*	
Mar 27, 2026				
	RAMTEK-ext, Vaibhav	State	changed state to Revision under Review at 12:30 pm *New Column added to the Reconciliation Retry tables as Payload as per Accenture Team requirement*	v64
Mar 23, 2026				
Revision in progress	RAMTEK-ext, Vaibhav	Edit	updated the page at 7:01 pm	

RAMTEK-ext, Vaibhav	State	changed state to	Revision in progress at 6:01 pm	v63