R0080 - IMPROVEMENTS TO 'FAILED TO DELIVER' CSS MESSAGES

ONBOARDING DOCUMENT

Version	Date	Author	Comments
1.0	03/06/2025	RTS	Initial RTS draft
1.1	10/06/2025	RTS	Amended following CSS Provider feedback.
2.0	24/06/2025	RTS	Amended following CSS Provider feedback.

BACKGROUND

R0080 – IMPROVEMENTS TO 'FAILED TO DELIVER' CSS MESSAGES

Since the CSS go-live, some messages have failed to reach their intended recipients. This is often due to gateway outages or message rejections. Once a message fails, the CSS Provider cannot resend it after the gateway is restored once standard re-try process has completed, there is no mechanism for this to be re-sent.

Currently, users lack visibility into messages that have failed to deliver.

This Change Proposal aims to enable a mechanism for resending of failed messages once a user's gateway is back online and to establish a sustainable solution for enabling visibility of messages users have rejected.

SOLUTION OVERVIEW

The solution introduces a mechanism for handling failed to deliver messages from the Central Switching Service (CSS) by:

- Creating a "Held Message Register" for messages that fail delivery after a 12-hour retry window.
- Providing an API for market participants (e.g., SDSPs and CSS Users) to request resends of these held messages.
- Using dashboards instead of static reports for real-time monitoring by authorized parties (DCC and Landmark).
- Notifying impacted participants of failed messages through DCC Service Desk (ServiceNow)

Overview of the Message Resend Process:

- Messages that fail delivery due to issues like endpoint downtime are stored.

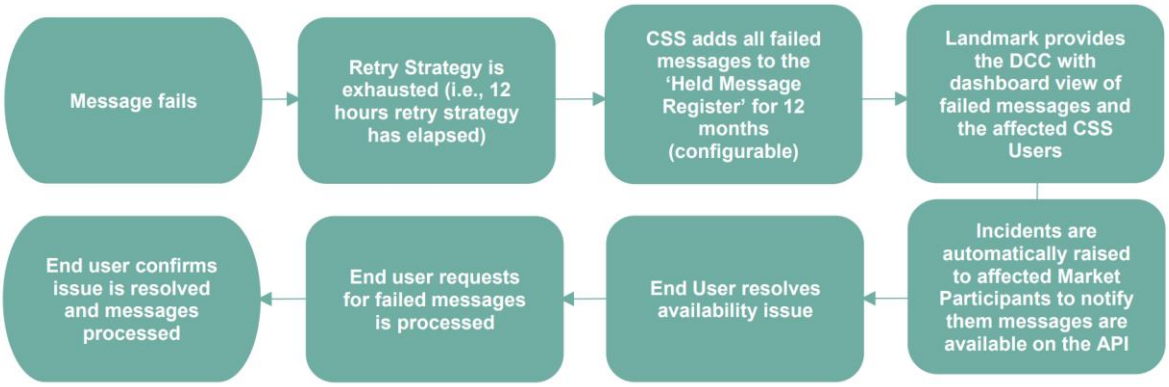
- Participants are notified of failed message instances (daily notification if messages have failed). The notification will be issued to parties Lead Contact on ServiceNow.
- Participants use the API to re-queue these messages.
- Messages are re-sent in chronological order (oldest first) to the participant's registered webhook.
- Messages will be retained within the Held Messages Register for 12 months, before removal.

Note: Use of the API is optional. However, if participants require the re-sending of failed to deliver messages, they will be directed to use the API.

Resend Queue Process Flow

The diagram below describes the high-level process for CSS messages which CSS receives the following HTTP response codes:

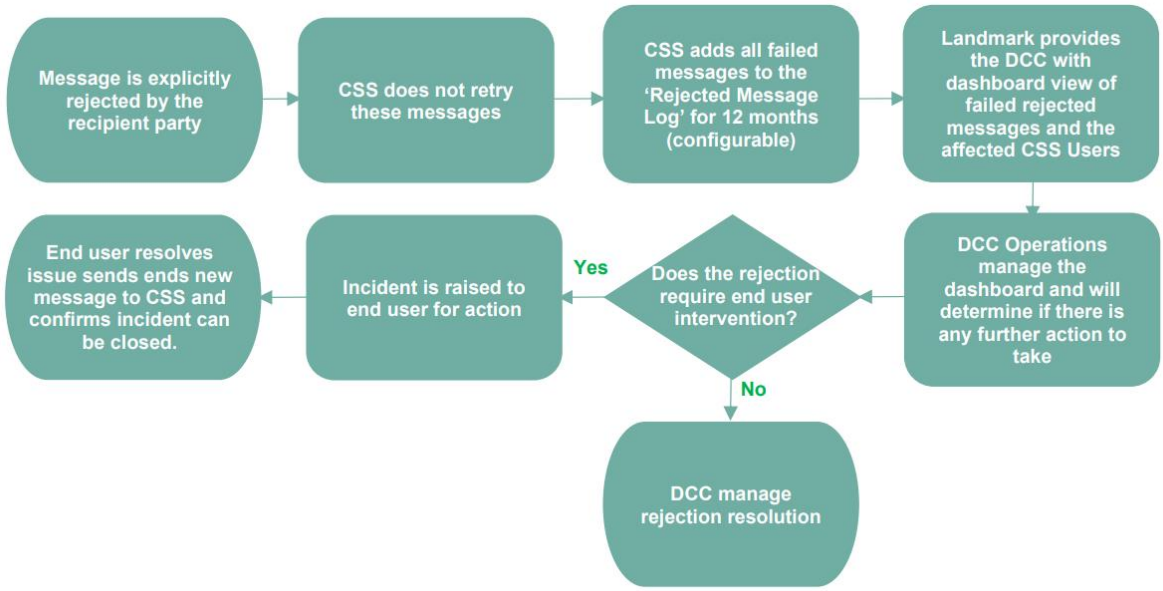
- 424 Failed Dependency
- 429 Too Many Requests
- 500 Internal Server Error
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout



Process for Rejected Messages (400 Response Codes)

The diagram below describes the high-level process for rejected messages where CSS receives a 400-response code.

Note: 424 and 429 error codes will fall into the resend queue process (see above). All other 400 codes will fall under the rejected messages process.



The technical specification for the API end-point called "Resend Held Messages" can be found below, or in Section 5.8 of the CSS interface Design Specification.

Message Id:
RECM_SN_CSS06030

Available to: Any market participant receiving messages from CSS.

Route: /system/messages/resend
Method: POST
Body:
{
"**role**": "string" --*
}

Responses:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}
This endpoint allows a market participant to request a re-send of messages that were not delivered after all retry attempts have been exhausted.
All documentation and information on R0080 can be found on the REC Portal, here.

R0080 USE CASES

Gateway Outage Recovery

- **Scenario:** A CSS User experiences a system outage, causing their gateway to be offline for over 12 hours (beyond re-try period). During this time, multiple messages from CSS fail to deliver.
- **Use Case:** Once the gateway is restored, the participant uses the API to request a resend of all failed to deliver messages from the outage window, ensuring no switching data is lost.

Certificate Expiry or Misconfiguration

- **Scenario:** A supplier's digital certificate expires unexpectedly, causing message delivery failures due to authentication errors.
- **Use Case:** After renewing the certificate and restoring connectivity, the supplier uses the API to retrieve the backlog of missed messages, avoiding manual incident tickets and delays.

High Volume Message Rejections

- **Scenario:** A CSS User deploys a new system update that inadvertently introduces schema validation issues, resulting in a large number of messages being rejected with 400 errors.
- **Use Case:** Visibility of rejected messages will be provided via a notification from ServiceNow.

Post-Incident Data Reconciliation

- **Scenario:** Following a major incident (e.g., a DCC-wide service disruption), a participant wants to ensure all messages were received and processed.
- **Use Case:** The participant uses the API to re-queue any failed to deliver messages, supporting accurate and complete data reconciliation.

Operational Monitoring and Proactive Recovery

- **Scenario:** A participant's operations team notices a drop in message volume via their internal monitoring tools.
- **Use Case:** The participant uses the API to access held messages and use the API to proactively retrieve them before any customer switching issues arise.

CSS SIMULATOR

- The CSS PIT Simulator is provided to CSS Users to simulate interactions with the CSS in their local (non-connected) environments. It is expected that CSS users utilise the CSS PIT Simulator in their Development environments to test their internal changes, rather than connecting to a test environment, unless driven by a REC Change Proposal.
- The CSS PIT Simulator will simulate the actions of CSS by responding to received messages and replying to messages replicating the CSS functionality.
- The CSS PIT Simulator enables parties to develop their applications and ensure that the messages they send to CSS conform to the CSS Physical Interface Design. This allows them to receive messages into their systems in the format that CSS will provide.

- The CSS PIT Simulator is available for both Windows and Linux platforms for all parties via GetOrganized, available at: https://dcc.getorg365.net/cases/DCC/DCC-2024-00003/default.aspx
- The supporting User Guide and Configuration guide can also be found in this location.
- CSS Users that require access to this GetOrganized location should email the following, providing the email address of the user requiring access: DCCSwitchingSI@netcompany.com
- The nominated user will be able to download, install and configure within their own environment.

INSTRUCTIONS

To implement the R0080 solution, the only requirement will be to configure access to the new Resend Held Messages API. This API is available to any Market Participant who receives messages from the CSS. The below information is included within the CSS Interface Design Specification v8.9.0

Message ID: RECM_SN_CSS06030
Route: /system/messages/resend
Method: POST
Body:
{
"role": "string" --*
}

Responses:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

As a pre-requisite action, Parties should confirm that all relevant CSS Production JWS Certifications are up to date and all relevant event webhooks are configured as expected. Set-up of the new Resend Held Messages API should not implicate any of the above.

Parties are not provided instructions in how calls to the Held Messages API are managed by their organisation, for example, a daily API-call can be made to see if any failed to deliver messages are received, or API calls can be made following the receipt of the Failed to Deliver Messages notification, which is issued to the Lead Contact via ServiceNow.

FAQ

1. Is the use of the Resend Held Messages API mandatory?

No. The API is optional. However, if a participant requests the re-sending of failed to deliver messages, they will be directed to use the API as the primary method.

---

2. What types of messages are stored in the Held Message Register?

Messages that fail delivery after the 12-hour retry window due to issues like endpoint downtime or transient system errors (e.g., HTTP 424, 429, 500–504) are stored in the register.

---

3. How long are messages retained in the Held Message Register?

Messages are retained for a configurable **Time To Live (TTL)** period, initially set to **12 months**. After this period, messages are no longer available for automated resend.

---

4. Can rejected messages (e.g., HTTP 400 errors) be resent using the API?

No. Messages rejected due to structural or business rule violations (HTTP 400) are not eligible for resend via the API. However, they are logged and visible via dashboards for operational review.

---

5. How are messages re-sent once requested?

Messages are re-queued in **ascending event date order** (oldest first) and sent to the participant's currently registered webhook for the relevant event type.

---

6. What happens if my webhook URL has changed?

Ensure your webhook URL is up to date in the CSS system before requesting a resend. Messages will be sent to the URL registered at the time of the resend request.

---

7. Can I still raise a manual ticket for message resend?

No.

---

8. How will I know if I have failed to deliver messages?

CSS will notify impacted participants via **Zendesk tickets**, which are routed through **DCC's ServiceNow** system.

---

9. What information is included in a re-queued message?

Re-queued messages include **original payload**, along with metadata such as **correlation ID**, **event ID**, and **event date**.

---

10. Where can I find the technical details of the API?

The API specification is available in **Section 5.8 of the CSS Interface Design Specification** and includes the endpoint, request format, and response codes.

---

11. Who should I reach out to if I am unable to connect to the API?

Participants who are struggling to receive held messages should **raise a ticket** with the CSS Service Desk.