Distributed Polling System
Error Handling Strategy

Version 1.0

Document Purpose

Purpose of this document is to design and formulate DPS error handling strategy at middleware layer to provide robust, reliable, guaranteed message delivery and easy maintenance.
We have segregated synchronous and asynchronous type of interfaces and we will specify treatment for each type of interfaces.

Some Basic Rules

1. A business process generally includes two or more applications.
2. Error handling strategy depicted here is at very high level.
3. By synchronous request from Web-DPS, we mean the whole business process is invoked synchronously from source.
4. By Asynchronous request from Web-DPS, we mean the whole business process is invoked asynchronously from source and within that business process there might be synchronous calls to each target application.
5. If member selects ‘SMS’ first preference then ‘Email’ second way of intimation and vice-versa.
6. Middleware won’t modify any data, If any failure occurs due to data (ex: corrupted data sent from source, mismatched data format between source and target application requirement and so on), it’s the application responsibility to modify and retrigger
7. Retry timing can be decided as per agreement
8. Technical Error is the error due to unavailability of the technical infrastructure like database failure, network down, server down and so on.
9. Functional error is the error due to mismatched expectation between applications - like target application expects a parameter as date-time format(11-11-2008 08:02:55) whereas source application sends only date(11-11-2008), or some new parameter has been configured in source application due to business needs whereas target application has not configured that parameter and causes unknown data at target system.
10. For synchronous requests (end-to-end I,e source to target via middleware) initiated from source, we assume if source has **X** time unit as time-out period, middleware should have **Y time-unit** for time-out (this includes both target API invoke) where **X > Y.**
    So middleware will be able to respond back to source within **X** time-unit.
11. Although we have depicted here the error handling strategy for Web-DPS→Middleware→SMS Gateway and Email server, where Web-DPS application is the originator, the other interfaces initiated from SMS Gateway and Email server, will be treated in the similar manner.
12. We have represented here target application shortly as ‘SMS Gateway/Email Server’, but these are two different applications and the error handling treatment is the same.

Table of Contents

Following (Section-1 and 2) are the scenarios, where there is synchronous invoke of middleware service using XML over HTTP or SOAP over HTTP from Web-DPS.

1. Synch Request from Web-DPS (Technical Error)

1.1 Web-DPS failed to invoke middleware service (Tech-Syn-001)

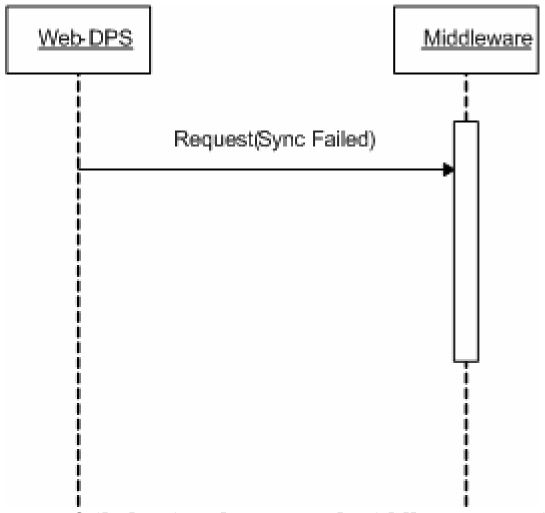

Figure 1-1: Web-DPS failed to invoke exposed middleware service

This is the scenario, if we finalize XML over HTTP or SOAP over HTTP from Web-DPS to middleware.

Failure Scenario	**Handling Strategy**
Web-DPS is unable to invoke exposed middleware service	• Web-DPS will retry for a predefined number of times with a specified interval. • If still unable to invoke, Web-DPS will raise an alert (through email) to maintenance support group for further action.

1.2 Middleware failed to invoke target system API (Tech-Syn-002)

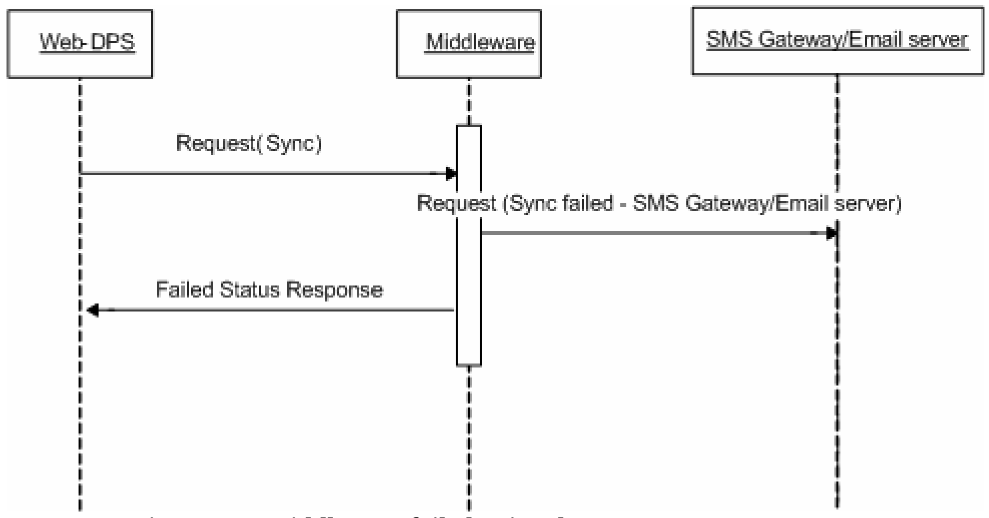

Figure 1-2: Middleware failed to invoke target system API

Failure Scenario	Handling Strategy
Middleware is unable to invoke any of the target system as per user's first preference - SMS Gateway or Email server API(Sync Request). *Info: User can specify in Web-DPS system as SMS first preference or Email as fisrt preference*	• As per business rules, middleware will initially invoke API of SMS Gateway or Email server (as per user's first preference). If middleware is unable to invoke the first target application, then middleware will invoke the second target application • Status of invoking 2nd target application will be responded back to Web-DPS application
Middleware is unable to invoke both SMS Gateway and Email server API(Sync Request).	• As per business rules, middleware will initially invoke API of SMS Gateway or Email server (as per user's first preference). If middleware is unable to invoke the first target application, then middleware will invoke the second target application. If still unable to invoke any of the two target applications, will raise an alert to support group through email. • Middleware will respond back to Web-DPS application as failed to invoke both target.

1.3 Middleware failed to receive response from target application (Tech-Syn-003)

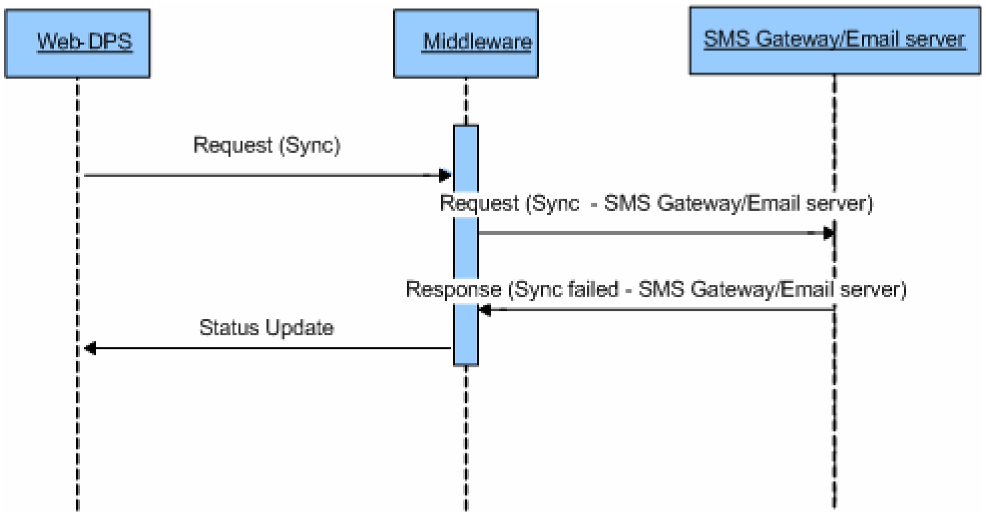
Web-DPS
Middleware
SMS Gateway/Email server
Request (Sync)
Request (Sync - SMS Gateway/Email server)
Response (Sync failed - SMS Gateway/Email server)
Status Update

Figure 1-3: Middleware failed to receive response from target

Failure Scenario	Handling Strategy
Web-DPS receives time-out response from middleware due to transaction-time out set at middleware layer. **Desc:** Web-DPS has sent a request to middleware and middleware has forwarded the request to first target application. But middleware does not receive response from first target application within transaction timeout period Specified at middleware layer. (might be due to network failure). Then middleware will retry (if designed that way) else will try for the second target application in the similar way. Eventually, middleware does not receive response from the second application too. **Steps:** **1. Web-DPS→Middleware** **2. Middleware→first Target Application** **3. Middleware does not receive response from first target app** **4. Middleware retries first target for few predefined configured number of times (if designed)** **5. Now middleware invokes 2nd target app, as it failed to inform customer through first target middleware→second target** **6. If middleware receives success from 2nd target, will respond back to Web-DPS** **7. else if does not receive response also from 2nd target, will retry, and once the time-out period set at middleware layer is reached , will respond back to Web-DPS.**	• Web-DPS will retry for this Sync request (as middleware intimated Web-DPS that from both target application middleware did not receive any response). • During retry, if the last request in target application was processed successfully, middleware will receive response as duplicate from target and will return SUCCESS to Web-DPS. **Note**: Need to verify how duplicate verification can be performed at each target application.

1.4 Web-DPS failed to receive response from Middleware (Tech-Syn-004)

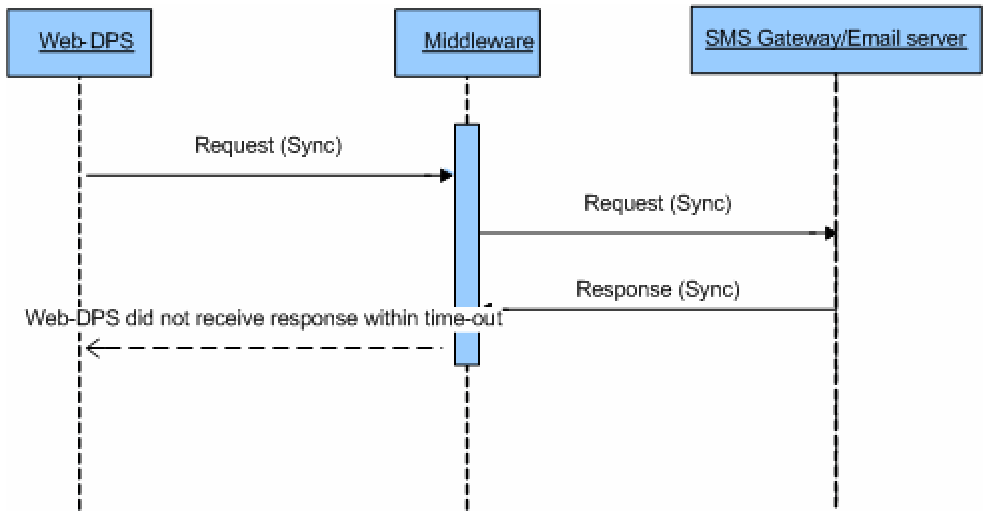

Figure 1-4: Web-DPS failed to receive response from Middleware

Failure Scenario	**Handling Strategy**
Web-DPS synchronous request timed-out, due to not receiving response from middleware (might be due to network failure). **Desc:** Web-DPS has sent a synchronous request to middleware and middleware has interacted with target applications. Whatever might be the response of middleware for handling target applications, middleware has responded back to Web-DPS. But Web-DPS has not received synchronous response (might be due to network failure).	• Web-DPS will retry (re-send the same request). • Middleware will maintain the status of previous request and using some key value, middleware will identify this as duplicate and will return back the actual response.

2. Synch Request from Web-DPS (Functional Error)

2.1 Web-DPS receives validation (Functional) error from middleware (Func-Syn-005)

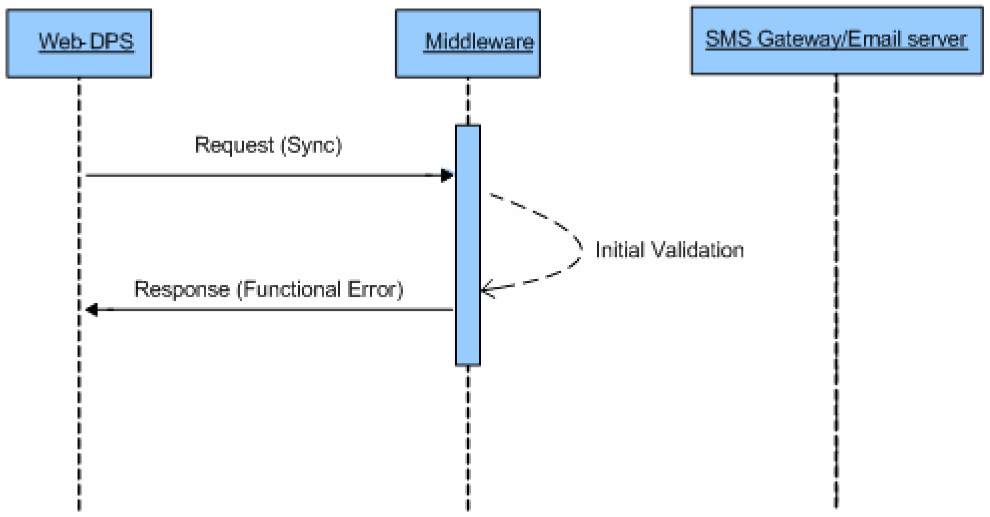

Figure 2-1: Web-DPS receives validation (Functional) error from middleware

Failure Scenario	Handling Strategy
Web-DPS receives response from middleware as functional error. **Desc:** Web-DPS has sent a request to middleware and after initial validation at middleware layer (like mandatory fields missing), middleware finds out validation error and responds back to Web-DPS as functional error.	• If modifications required at Web-DPS end on data, poll creator can modify the data using Web-DPS screen and resend to middleware for re-processing. • If any modifications required at middleware end, middleware will update accordingly and later inform Web-DPS to resend the request.

2.2 Web-DPS receives functional error from target application (Func-Syn-006)

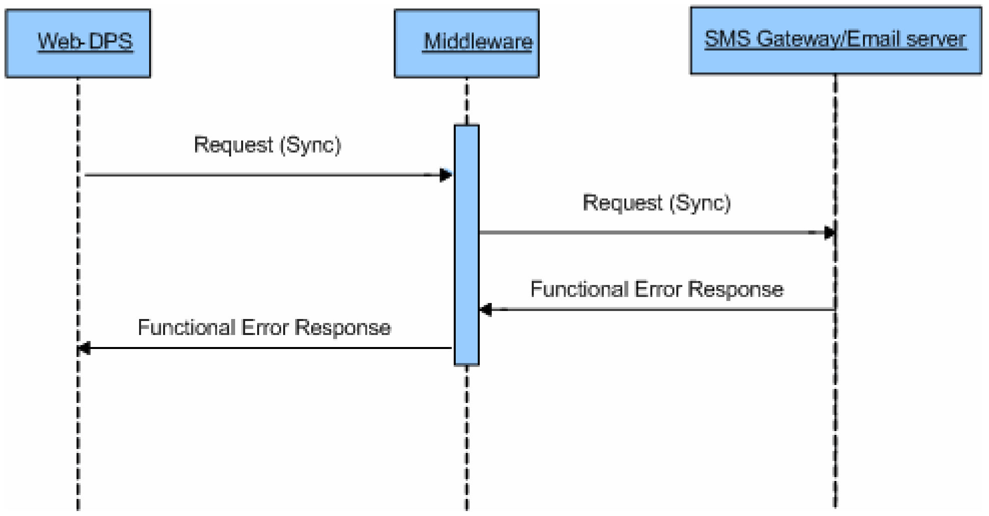

Figure 2-2: Web-DPS receives functional error from target application

Failure Scenario	**Handling Strategy**
Web-DPS receives functional error response returned by target application. **Desc:** Web-DPS has sent a request to middleware and middleware has forwarded to target applications. But target applications returns functional error and middleware responds back to Web-DPS as functional error.	• If modifications required at Web-DPS application end, poll creator can modify the data using Web-DPS screen for all functional error out transactions and resend to middleware for re-processing. • If any modifications required at target application end, target application will update accordingly and later inform Web-DPS to resend the request.

If we choose JDBC adapter communication approach, then middleware needs to communicate with Web-DPS application database with pull mechanism (called database polling), instead of Web-DPS pushing the interface.

3. Async Request from Web-DPS (Technical Error)

3.1 Middleware failed to connect to source application (Tech-Asyn-007)

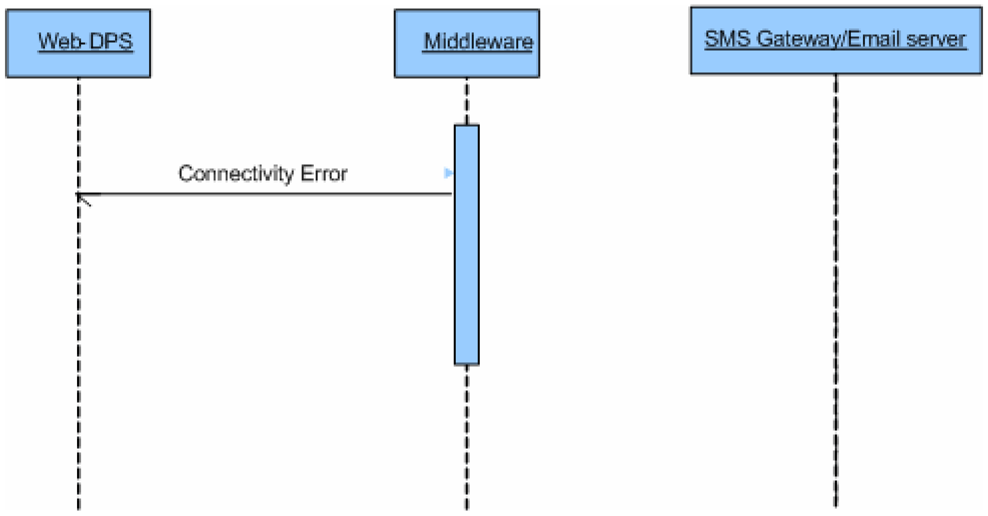

Figure 3-1: Middleware failed to connect to Web-DPS

Failure Scenario	**Handling Strategy**
Middleware is unable to connect to Web-DPS database to process request. **Desc:** Web-DPS has triggered an asynchronous request to middleware-polling-table residing in Web-DPS database and middleware is unable to connect to Web-DPS database to process that request.	• Middleware will retry for a predefined number of times with a specified interval. • After the maximum number of retries from middleware, middleware will raise an alert (email) to support group for further action to be taken.
Middleware is unable to connect to Web-DPS database to archive a request (middleware has already picked up from Web-DPS database). **Desc:** Web-DPS has triggered an asynchronous request to middleware-polling-table residing in Web-DPS database and middleware has picked up that request from Web-DPS database to process that request. Now after initial validation of the request message at middleware layer, while going to archive that event at Web-DPS database, middleware is not able to connect.	• Middleware will retry for a predefined number of times with a specified interval. • After the maximum number of retries from middleware, middleware will raise an alert (email) to support group for further action to be taken. • Middleware will proceed to interact with target applications (SMS Gateway and Email).

3.2 Middleware failed to connect to target (Tech-Asyn-008)

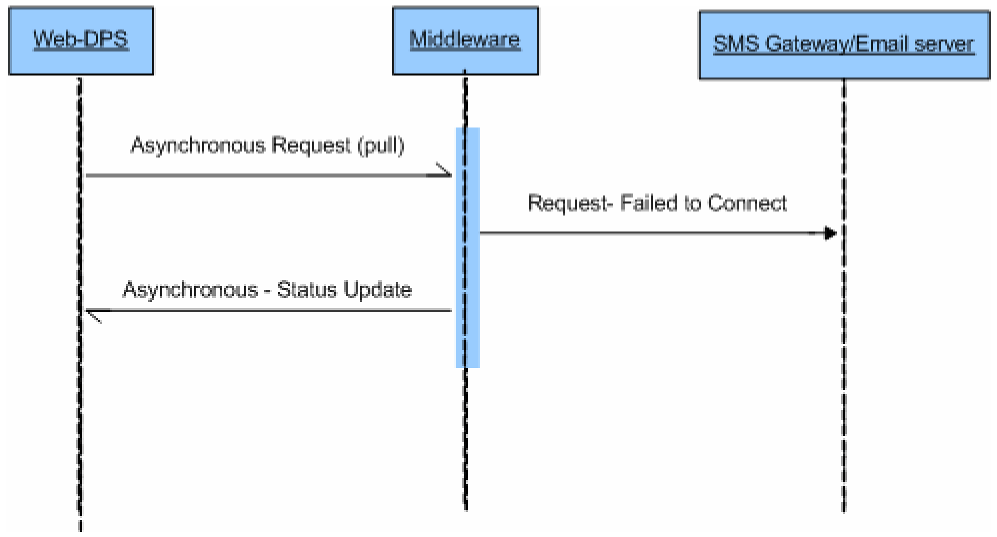
Web-DPS
Middleware
SMS Gateway/Email server
Asynchronous Request (pull)
Request- Failed to Connect
Asynchronous - Status Update

Figure 3-2: Middleware failed to connect to target application

Although the whole business process is asynchronous, still the call to SMS Gateway or Email server is synchronous.

Failure Scenario	Handling Strategy
Middleware is unable to invoke any of the target application as per user's first preference - SMS Gateway or Email server API. *Info: User can specify in Web-DPS system as SMS first preference or Email as first preference*	• As per business rules, middleware will initially invoke API of SMS Gateway or Email server (as per user's first preference). If middleware is unable to invoke the first target application, then middleware will invoke the second target application • Status of invoking 2nd target application will be updated back at Web-DPS application asynchronously.
Middleware is unable to invoke both SMS Gateway and Email server.	• As per business rules, middleware will initially invoke API of SMS Gateway or Email server (as per user's first preference). If middleware is unable to invoke the first target application, then middleware will invoke the second target application. If still unable to connect to the two target applications, will raise an alert (email) to support group through email. • Middleware will update status in Web-DPS application database as failed to connect to both target. • middleware needs to store the event for future resubmission • Once the problem is rectified, the event needs to be resubmitted from middleware layer

3.3 Middleware failed to receive response from target application (Tech-Asyn-009)

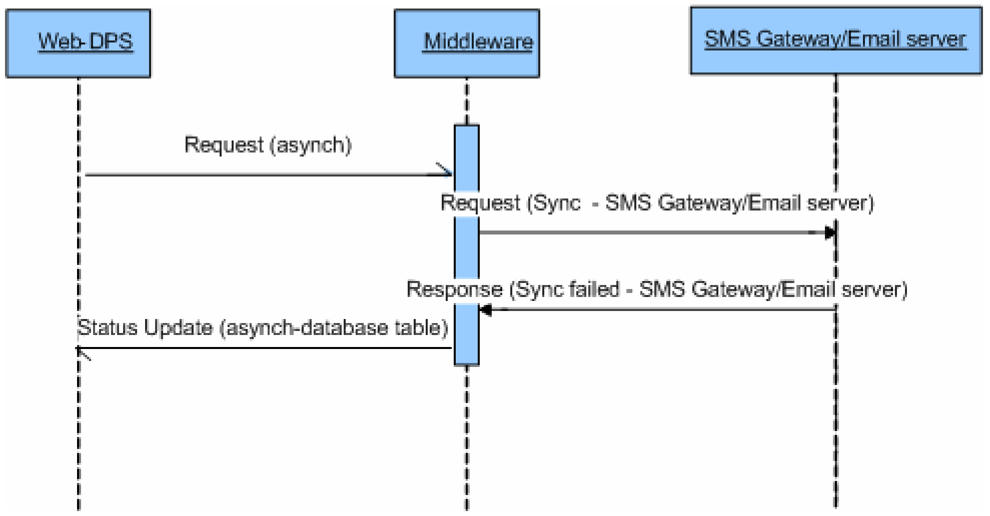
Web-DPS
Middleware
SMS Gateway/Email server
Request (asynch)
Request (Sync - SMS Gateway/Email server)
Response (Sync failed - SMS Gateway/Email server)
Status Update (asynch-database table)

Figure 3-3: Middleware failed to receive response from target

Failure Scenario	Handling Strategy
Web-DPS database table (Status Column) is updated with 'Failed to receive response from both SMS Gateway and Email' **Desc: M**iddleware has picked up an event from Web-DPS database table and after initial validation, forwarded the same to first target application (as per member's first preference). But middleware does not receive response from first target application within transaction timeout period Specified in middleware. (might be due to network failure). Then middleware will retry (if designed that way) else will try for the second target application in the similar way. Eventually, middleware does not receive response from the second application too. **Steps:** **8. Web-DPS→Middleware (asynch)** **9. Middleware→first Target (synch) Application** **10. Middleware does not receive response from first target app** **11. Middleware retries first target for few predefined configured number of times (if designed)** **12. Now middleware invokes 2nd target app, as it does not know the status of intimating member through first target application middleware→second target (sync)** **13. Again, if it does not receive response also from 2nd target, will retry, and once the time-out period set at middleware layer is reached,**	• As middleware is unaware of the request status from both the application, middleware needs to store that event for future resubmission. • Middleware will raise an alert (email) to support group • Once the problem is rectified, the event needs to be resubmitted from middleware with the initial preferred way of communication medium.

will update status back at Web-DPS.	
Web-DPS database table (Status Column) is updated with 'Intimated member through second application medium' **Desc: M**iddleware has picked up an event from Web-DPS database table and after initial validation, forwarded the same to first target application (as per member's first preference). But middleware does not receive response from first target application within transaction timeout period Specified in middleware. (might be due to network failure). Then middleware will retry (if designed that way) else will try for the second target application in the similar way. Eventually, middleware succeeds to intimate member through second preferred medium of communication. **Steps:** 14. **Web-DPS→Middleware (asynch)** 15. **Middleware→first Target (synch) Application** 16. **Middleware does not receive response from first target app** 17. **Middleware retries first target for few predefined configured number of times (if designed)** 18. **Now middleware invokes 2nd target app, as it failed to intimate member through first target middleware→second target (sync)** 19. **Now, middleware receives success from 2nd target, updates Web-DPS database table status column accordingly (asynchronously)**	• Middleware does not need to store this event for resubmission as it's a success • Updates the Web-DPS table accordingly

3.4 Middleware failed to update response at Web-DPS (Tech-Asyn-010)

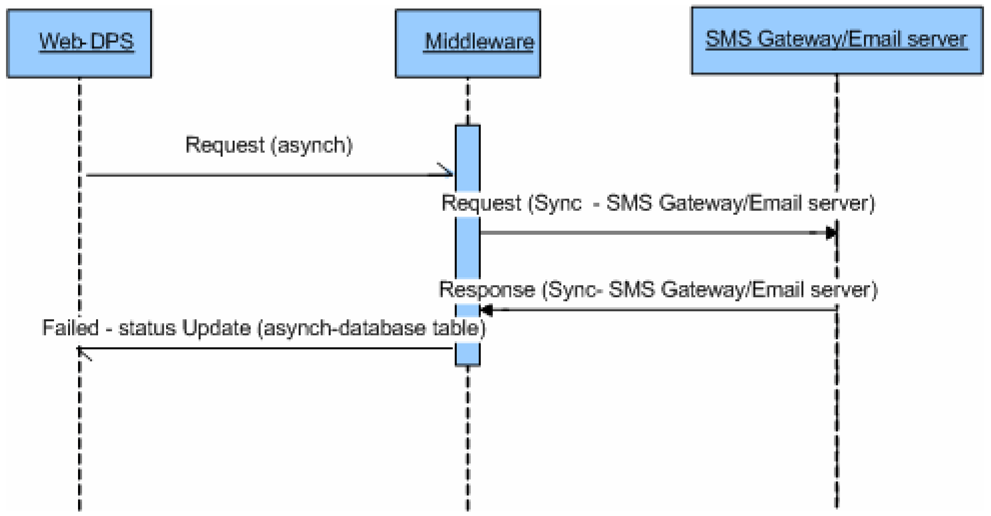
Web-DPS
Middleware
SMS Gateway/Email server
Request (asynch)
Request (Sync - SMS Gateway/Email server)
Response (Sync- SMS Gateway/Email server)
Failed - status Update (asynch-database table)

Figure 3-4: Middleware failed to update response at Web-DPS

Failure Scenario	Handling Strategy
Middleware failed to update final response back at Web-DPS database table (might be due to network failure). **Desc:** Middleware has picked up an event from Web-DPS database table and interacted with the target applications (as per member's preference). But while middleware is updating the final status back at Web-DPS, it failed (might be due to network failure)	• Middleware will retry. • Middleware needs to store that event for future resubmission. • Middleware will raise an alert (email) to support group • Once the problem is rectified, the same event needs to be resubmitted from middleware layer.

4. Async Request from Web-DPS (Functional Error)

4.1 Siebel receives validation (Functional) error from middleware (Func-Asyn-011)

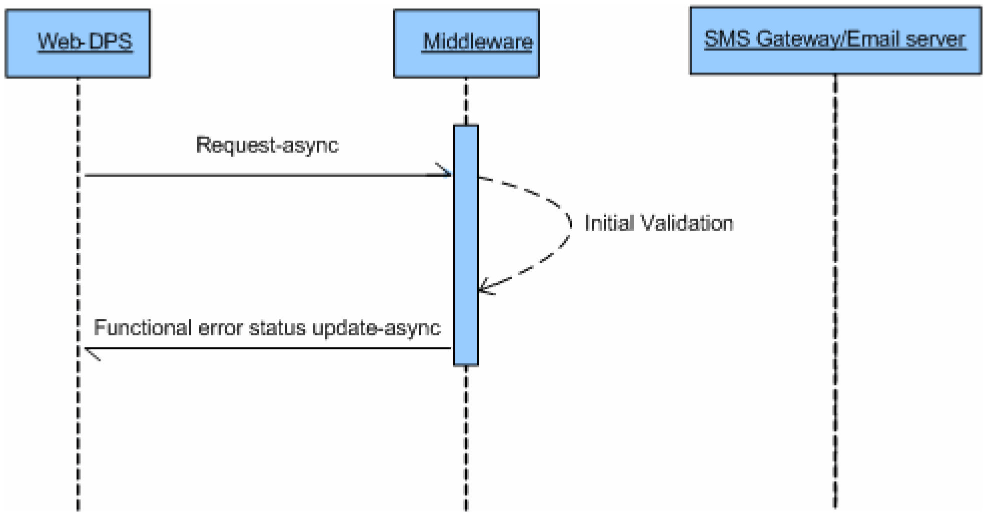

Figure 4-1: Web-DPS receives validation (Functional) error from middleware

Failure Scenario	Handling Strategy
Web-DPS database table (status column) updated with the response from middleware as functional error. **Desc:** Middleware has picked up an event from Web-DPS database polling-table and after initial validation at middleware layer (like mandatory fields missing), middleware finds out validation error. Middleware updates archive table status column as functional error.	• If modifications required at Web-DPS end, poll creator can modify the data using Web-DPS screen and resend to middleware for re-processing. • If any modifications required at middleware end, middleware will update accordingly and later inform Web-DPS to resend the request.

4.2 Web-DPS receives functional error from target application (Func-Asyn-012)

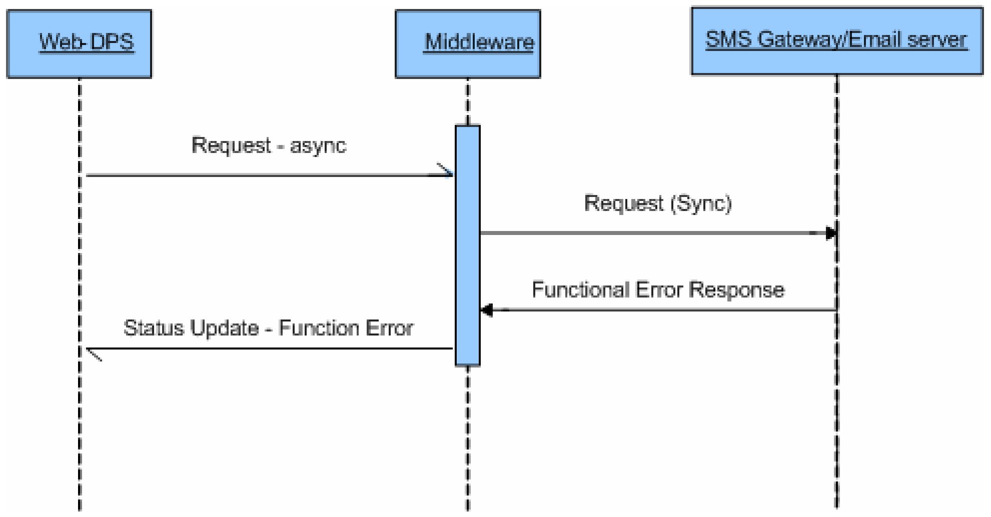

Figure 4-2: Web-DPS receives functional error from target application

Failure Scenario	**Handling Strategy**
Web-DPS receives functional error response returned by target application. **Desc:** Middleware has picked up an event from Web-DPS database polling-table and after initial validation at middleware layer (like mandatory fields missing), middleware forwards the request to target application. Target application returns functional error and middleware updates the same at Web-DPS application archive database table (status column).	• If modifications required at Web-DPS application end, poll creator can modify the data using Web-DPS screen for functional error out transaction and resend to middleware for re-processing. • If any modifications required at target application end, target application will update accordingly and later inform Web-DPS to resend the request.