

Project Maverick – Plan to  
Manufacture  
Functional Lean Specification

Interface – Break Fix Interface from  
Maestro to Oracle -  
(PTM\_INV\_INT\_146)

## Document Control Information

### Document Information

<b>Document Identification</b>	PTM_INV_INT_146
<b>Document Name</b>	Break Fix Interface from Maestro to Oracle
<b>Project Name</b>	Project Maverick
<b>Client</b>	Dell
<b>Document Author</b>	Deloitte Project Maverick PTM Team
<b>Document Version</b>	V1.0
<b>Document Status</b>	Draft
<b>Date Released</b>	01-XX-2025

### Document Edit History

<b>Version</b>	<b>Date</b>	<b>Additions/Modifications</b>	<b>Prepared/Revised by</b>
1		Initial version	Deloitte Project Maverick PTM Team

### Document Review/Approval History

<b>Date</b>	<b>Document Version</b>	<b>Name</b>	<b>Organization/Title</b>	<b>Comments</b>
< dd-mmm-yyyy >		Vasu Hariharan	Dell	< dd-mmm-yyyy >
		Priya Udayashankar	Dell	
		Jeffrey Coger	Dell	
		Steve Jennings	Dell	

### Distribution of Final Document

The following people are designated recipients of the final version of this document:

<b>Name</b>	<b>Organization/Title</b>
<Name>	<Organization/Title>

---

# Table of Contents

- 1. Summary . . . . . 4
  - 1.1 Purpose/Justification . . . . . 4
  - 1.2 Interface Description and Overview . . . . . 4
  - 1.3 Business Rules . . . . . 5
  - 1.4 Sample Output . . . . . 6
  - 1.5 Definitions and Acronyms . . . . . 6
- 2. Functional Design . . . . . 7
  - 2.1 Interface Details . . . . . 7
  - 2.2 Launch Parameters . . . . . 8
  - 2.3 Key Logic . . . . . 8
  - 2.4 Data Mapping . . . . . 10
  - 2.5 Validation & Error Handling Requirements . . . . . 19
  - 2.6 Notification Requirements . . . . . 11
- 3. Functional Unit Test Cases . . . . . 20
- 4. Open and Closed Issues . . . . . 21
  - 4.1 Open Issues . . . . . 21
  - 4.2 Closed Issues . . . . . 21
- 5. Appendix . . . . . 22

---

## 1. Summary

As part of Project Maverick, Dell Technologies is embarking on a journey to move from legacy Prism Oracle EBS ERP system and onboarding Oracle Cloud ERP as their new ERP system to handle Finance and Supply Chain processes. As part of this project, there will need to be interfaces set in place for Oracle Cloud ERP to communicate with continuing legacy applications such as Maestro. The Break-fix Interface from Maestro to Oracle Cloud is designed to streamline and automate the inventory and sub-inventory process, ensuring that stock levels are maintained efficiently and accurately as well as automate the process of updating inventory records between the two systems. This interface facilitates the seamless transfer of transaction level data from the Maestro system to Oracle Cloud, leveraging the strengths of both platforms to optimize inventory management.

The primary objective of this interface is to ensure that break-fix orders, especially the shipments and returns generated in Maestro are accurately reflected in Oracle Cloud. This involves the extraction of relevant data from Maestro, transformation of the data into a format compatible with Oracle Cloud which is JSON in this case, and the subsequent loading of this data into the Oracle Cloud system. The interface will capture inventory break-fix adjustments, returns and additional information. This will ensure that all necessary information, such as item details, quantities, and order dates, is accurately captured and transferred.

### 1.1 Purpose/Justification

The purpose of the breakfix Interface from Maestro to Oracle Cloud is to streamline the inventory management process, ensuring that inventory levels are maintained efficiently and accurately between service orgs and Maestro. This interface facilitates the seamless transfer of break-fix shipment and return transactions between the Maestro system and Oracle Cloud, reducing manual intervention and minimizing the risk of errors. These service facilities will need to move materials between organizations due to demand. Maestro will be the application who analyzes the service part requests and orchestrates this material movement across the 3PL locations. The interface will come into play when there is a break-fix shipment or break-fix return transaction from Maestro. Oracle will need to communicate with Maestro to know these transactions and then update its own system as Oracle will be the source of all inventory management in the future state. By integrating these systems, Oracle can achieve real-time visibility into inventory levels, optimize stock replenishment cycles, and enhance overall supply chain efficiency.

Overall, the Interface from Maestro to Oracle Cloud represents a significant enhancement to the inventory management capabilities of an organization, providing a reliable and efficient means of maintaining optimal stock levels and supporting the smooth operation of supply chain processes.

### 1.2 Interface Description and Overview

The breakfix Interface from Maestro to Oracle Cloud is designed to establish a connection between these two applications regarding the break-fix process as well as inventory modifications. This interface ensures that inventory levels are maintained with seamless transfer of break-fix data and any other changes within inventory data from the Maestro system to Oracle Cloud for part change and sub-inventory transfer.

The interface operates by extracting relevant data from Maestro, such as shipment and return from DLP orgs to Float Orgs to Oracle. This data is then transformed into a format that is compatible with Oracle Cloud, ensuring that it can be accurately and efficiently processed by the receiving system.

---

Once the data is sent, Oracle Cloud will process the received data either into a break-fix shipment or break-fix return and update the received fields accordingly. This automated process reduces the need for manual intervention, minimizing the risk of errors and ensuring that inventory levels are consistently maintained.

Below systems will be involved in the overall process of the transmission -

1. Maestro - Dell owned system which will be involved in the initiation of services break-fix and service inventory adjustment processes
2. Kafka – the queuing system in the middleware to ensure efficient processing of bulk messages through proper allocation of resources
3. Oracle Integration Cloud Service (OIC) – the middleware integration layer system that will replace the current legacy Prism Eagle middleware system and will hold the transformation and orchestration logic to drive integration flows.

### 1.3 Business Rules

The Interface from Maestro to Oracle Cloud is designed to ensure seamless and efficient inventory management. The following business rules are essential for the successful implementation and operation of this interface:

1. Data Accuracy: All inventory data transferred from Maestro to Oracle Cloud must be accurate and up to date. This includes item quantities, descriptions, and locations. Any discrepancies must be flagged and resolved promptly to maintain data integrity. Maestro will also include data from HES inventory in the future state. There will need to be recognition of this type of inventory when Oracle is interfaced with Maestro.
2. Real-Time Synchronization: The interface must support real-time updates to reflect the current inventory status. This ensures that any changes in inventory levels, such as new stock arrivals or items issued, are immediately reflected in Oracle Cloud.
3. Error Handling: Robust error handling mechanisms must be in place to identify and address any issues during data transfer. This includes logging errors, notifying relevant personnel, and providing clear instructions for resolving the issues.

#### Assumptions.

S.no.	Assumption
1	Maestro will orchestrate material movement requests across 3PL locations
2	The messages from Maestro will come in the Kafka queue for the integration
3	There will be no direct communication between OIC and Maestro. All communication will be through Kafka
4	Any transaction processing within Maestro systems is out of scope and this integration will cover only the receiving and processing of messages from Maestro
5	Xsd message specs is not required as the payload is in JSON format.
6	UOM for any item is not included in the xml structure. It is assumed that the UOM is EA by default for all transactions
7	All prerequisite configs like inv orgs, sub-inventories, locations, Account Aliases and DFFs should be configured prior to invoking this interface

---

By adhering to these business rules, the Interface from Maestro to Oracle Cloud will provide a reliable and efficient solution for managing inventory, ensuring that users can maintain optimal stock levels and streamline their operations.

**1.4 Sample Output**

**1.5 Definitions and Acronyms**

Acronym	Meaning	Description
ERP	Enterprise Resource Planning	A type of software used by organizations to manage business activities.
API	Application Programming Interface	A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.
PaaS	Platform as a Service	A set of Cloud services that delivers infrastructure and middleware components in the cloud that enables developers and IT administrators to build and manage mobile apps and web applications.

---

## 2. Functional Design

### 2.1 Interface Details

Encryption / Decryption - Data Security	No
Date/time format alignment on Payload	2024-12-10T09:00:02.000+00:00
Sequence of Processing	Kafka to OFC
Determination of if the interface will be trigger/Invoke or both	trigger
Full Payload Processing or Partial Processing Allowed	Full Processing
Frequency - Real Time/ Near Real Time/ Batch	Real Time
Volume	80K Per day
Payload Size	

The interface will operate on a scheduled basis, typically running at predefined intervals to ensure timely updates. The data flow will include the following key elements:

1. Source System (Maestro):

- The Maestro system will generate sub-inventory transfer and inventory adjustment data based on transaction happening in Maestro, inventory levels and org communications. Required transactions will be done as the same as well as cooperate with predefined rules.
- Data elements to be extracted include order type, item details, quantity required, source and destination org and sub inventory information.

2. Data Extraction:

- A data extraction process will be implemented to pull the necessary sub-inventory transfer and inventory adjustment data from Maestro. This process will ensure that only relevant and up-to-date information is extracted.
- The extraction process will include validation checks to ensure data accuracy and completeness.

3. Data Transformation:

- Extracted data will undergo transformation to match the data structure and format required by Oracle Cloud which is JSON format.
- Transformation rules will be defined to map data fields from Maestro to the corresponding fields in Oracle Cloud.

4. Data Loading:

- Transformed data will be loaded into Oracle Cloud using the appropriate APIs or data import mechanisms.
- The loading process will include error handling and logging to ensure successful data transfer and to identify any issues that may arise.

5. Target System (Oracle Cloud):

- Oracle Cloud will receive the sub-inventory transfer and inventory adjustment data and update its inventory records accordingly.
- The system will trigger any necessary workflows or notifications based on the updated inventory data.

By implementing this Interface, organizations can achieve greater efficiency in inventory

---

management, reduce the risk of stockouts, and ensure that inventory levels are optimized to meet demand.

## 2.2 Launch Parameters

Parameter	Required	Valid Values	Default

The launch parameters for this Interface inbound to Oracle from Maestro do not have any explicit parameters required.

## 2.3 Key Logic

### Shipment

- The logic for shipment is that we move the inventory from DLP org to Float org.
- The destination Org Code is fetched from “Ship-to” location”
- Destination Subinventory is always “Usable”
- MessageTypeID would be “PartsStatusUpdateShipped”
- For break-fix, we need to create in-transit shipment. Once the in-transit shipment is created, it needs to be received at the destination immediately.
- We need to capture fields like Source Code, DPS Number, Transaction Reference is NPR Reason, Ship Date, Waybill Number
- Transaction date is defaulted to “sysdate”
- We also need to capture OOW, KYC, KYHD, WUD1 and Bill To flag values in Attribute 12
- For float management, we need to store the DPS number (coming from attribute 16 of transaction history DFF) as Lot number in the destination float org.
- For Float Management, we need to store Part Number, Float Org, DLP org and sub inventory, Quantity, Serial Numbers, Lot Number (DPS Number)
- If another transaction comes in with the same combination of Item and Lot details, The write-off needs to happen based on the earliest date.

### Break-fix return:

- The main columns we need to identify would be Action (which would be “returned”), DPS Number, Line Number, Item Shipped and Item received, Quantity received, Dest Org and Dest Subinventory. If Item shipped is Null, it need to be defaulted to Item received.
- MessageTypeID would be “PartsStatusUpdateReturned”
- We need to use LOT Number (which is Dispatch ID field) and item number combination to identify the Float Org and Subinventory.
- In All of the below scenarios, we need consider serial numbers if serial numbers are available while doing the respective activity.
- If the serial numbers are different for both Shipment and Return transactions, then, first the existing Serial Numbers need to be issued out using Account Alias “Part Change” and then do Account Alias receipt with New Serial Numbers with Account Alias “Part Change”
- If the Serial Number which needs to be issued out is not mentioned, issue out the serial number in the FIFO format.
- There are multiple cases for breakfix return and below are the scenarios:



---

**Case1:**

- ➔ When Item shipped is same as Item received in return payload and when the Lot number exists in the float org with no write-off, this is considered as a happy path scenario.
- ➔ In this case, we first need to identify whether the item and LOT combination that we received as part of return payload were also received as part of shipment. To check if they were received as part of shipment, we need to check the on-hand table for item and lot-combination. If the combination exists, we need to do the in-transit shipment and receipt of item from Float Org to DLP Org for the Item which is there in the Item Received field.

**Case2:**

- ➔ When Item shipped is same as Item received in return payload and when the combination of item and LOT number doesn't exist in float org that were received as part of shipment payload, we need to check if it is a case of write-off or not. This means the combination of LOT and Item number doesn't exist in the on-hand table.
- ➔ If it is not write-off, this means that the LOT and item combination doesn't exist in on-hand table, because it can be of some other issue or consumption transaction but not a write-off transaction, we need to do Account Alias receipt at the DLP Org (shipment source org) by using the Item received value and the Alias would be "Additional Exchange Receipts".
- ➔ If it is because of write-off that the LOT and item combination doesn't exist in on-hand table because of a write-off transaction. Then, first, we do Account Alias Receipt at the float org, and we use the same Alias that was used while writing off and we need to use item received for this transaction. Then, do an in-transit shipment from float org to DLP Org for the Item Received. If the serial numbers are different between shipment and return transactions, first, we need to issue out the older serial numbers using "Part Change" account Alias and bring in new serial numbers with "Part Change" account Alias receipt. We need to consider the serial number for this process.

**Case3: Part Change**

- ➔ If Item shipped and Item Received is not the same and lot and item combination exists in the float org for shipped item, and item shipped and transaction qty is less than or equal to open float qty, then the transaction is a part change quantity
- ➔ In this case, Account Alias issue need to be performed with the item shipped at the float org with the serial number that was shipped and the account alias is "Part Change" and we need to do Account Alias receipt with the Item Received at Float Org with the serial number and the account alias is "Part Change". These 2 should happen parallelly.
- ➔ The Second step for this case would be a in-transit shipment and receipt from float to DLP Org for Item Received.

**Case4: Part Change + WriteOff Reversal:**

- ➔ If shipped item and received item are not the same on the return payload and if item shipped and Lot combination doesn't exist due to prior write-off
- ➔ In this case, first we need to Account Alias Receipt for Item shipped at the Float Org and the Alias and serial number should be same as the one used for Write off
- ➔ The second step is to do Account Alias Issue for shipped Item and serial number at the float Org with Account Alias as "Part Change" and Account Alias Issue for Item Received at Float Org with Alias as Part Change. Both these transactions need to go in Parallel.
- ➔ The last step is to do a in-transit shipment and receipt from Float to DLP Org for the Item received.

**Case5: RNS**



- ➔ If Breakfix return comes without a breakfix shipment (or) If breakfix return is sent post 180 days of Breakfix Shipment (NON PFR) or If Breakfix Return is sent post 360 days of Breakfix Shipment (PFR)

- In this case, Account Alias Receipt needs to happen at DLP Org for Item Received with Alias as "RNS"

## 2.4 Data Mapping

SNO	Report Heading	Table	Column/Field Name

This Interface will be using the same standard payload for each scenario explained above, different fields will be visible. Refer to the Sample Output section to view the fields present for break-fix shipment and return. The following is a .JSON for the standard payload of the shipment and return interface:

   
PTM\_INV\_INT\_146\_P PTM\_INV\_INT\_146\_P  
artStatusUpdateShipp artStatusUpdateRetur

The data mapping for the break-fix Interface from Maestro to Oracle Cloud involves defining the specific data elements that will be transferred between the two systems. This process ensures that the data is accurately and consistently translated from the source system (Maestro) to the target system (Oracle Cloud).

The following tables explain which fields will be captured between shipment and return:

Break-Fix Shipment		
Field Name to Capture	Oracle Mapping	Attribute # if applicable
Delivery Details Status	Item Number	
Quantity	Quantity	
Message Value	Source	
Transaction Reference (NRP Reason)		
Waybill Number	WaybillNumber	
Line Number	Line	
NOTA Fiscal		
PPID	Serial Number	
Dispatch ID	DPS and LOT	Attribute 16
PartID	Item Number	
Source	Source Org	
SourceInventoryName	Source Sub Inventory	
Destination	Destination Org	
NRP_REASON		Attribute 12
OOW_FLAG		Attribute 12
KYC_FLAG		Attribute 12
KYHD_FLAG		Attribute 12
WUD1		Attribute 12
Bill To		Attribute 12

Break-fix Return		
Field Name to Capture	Oracle Mapping	Attribute # if applicable
PartID	Item Number	
Quantity	Quantity	
Type: SourceInventoryName	destination sub inventory	
SerialNumber	Standard serial number	
LocationID	Inventory Org	
Type: Increase/Decrease	Increase is Account Alias Receipt	
DispatchID	DPSNumber	
Adjustment Number	Partner Transaction ID	
RootCause	Root Cause	
Reason	Reason	

## HES Debrief:

For ISG related transactions, we may have 2 more message types coming in from Maestro.

The 2 Message types are:

1. PartsUsageUpdate
2. ServiceCallClosure

For both these Message types, we need to perform the below account alias transactions based on the combination of "Debrief Type", "Disposition Type", "Disposition Reason".

All the transactions need to be performed on the float org of the shipment transaction and in the "USABLE" Sub-inventory.

S. No	Debrief Type	Disposition Type	Disposition Reason	Action to be performed	Transaction Type and Logic	DFFs
1	INSTALL	INSTALLED		INSTALL Transaction	Account alias issue from Float Org. Account Alias is "Field Service Usage"	<b>Source:</b> For transaction 1- l_work_order  '- '  l_record_id  '- '  'INSTALL'  <b>Reference:</b> MAESTRO- PART_USAGE  <b>Serial Number Attributes:</b> 1. Attribute1=ReturnW

						aybillNumber
2	REMOVE	LEFT ONSITE	PART RETENTION	REMOVE followed by Part Retention Issue	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery" followed by Account Alias Issue with Alias name "Site Adjustment"	<p><b>Source:</b> For Transaction 1- L_work_order  '- '  L_record_id  '- '  'REMOVE'</p> <p><b>Reference</b> MAESTRO- PART_USAGE</p> <p>For Transaction 2- <b>Source:</b> transaction_source _name = L_work_order  '- '  L_record_id  '- '  'PR_ISSUE'</p> <p><b>Reference</b> MAESTRO- PART_USAGE</p> <p><b>Serial Number Attributes:</b> 1. ReturnWaybillNumber for first transaction</p>
3	REMOVE	LEFT ONSITE	DISK ERASURE	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<p><b>Source:</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'</p> <p><b>Reference</b> MAESTRO- PART_USAGE</p> <p><b>Serial Number Attributes:</b> 1. ReturnWaybillNumber</p>

4	REMOVE	RETURNED TO WAREHOUSE	DEFECTIVE	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> transaction_source_name = L_work_order  '-' '  L_record_id  '-' '   'REMOVE'
						<b>Reference</b> MAESTRO-PART_USAGE
						<b>Serial Number Attributes:</b> 1. ReturnWaybillNumber
5	REMOVE	RETURNED TO WAREHOUSE	CUSTOMER FA	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '-' '  L_record_id  '-' '   'REMOVE'
						<b>Reference</b> MAESTRO-PART_USAGE
						<b>Serial Number Attributes:</b> 1. ReturnWaybillNumber 2. FA_FLAG = "Y"
6	REMOVE	RETURNED TO WAREHOUSE	MANDATORY FA	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '-' '  L_record_id  '-' '   'REMOVE'
						<b>Reference</b> MAESTRO-PART_USAGE
						<b>Serial Number Attributes:</b> 1. ReturnWaybillNumber 2. FA_FLAG = "Y"

7	REMOVE	LEFT ONSITE FOR PICKUP	DISK ERASURE	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumber
8	REMOVE	LEFT ONSITE FOR PICKUP	CUSTOMER FA	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumber 2. FA_FLAG = "Y"
9	REMOVE	LEFT ONSITE FOR PICKUP	MANDATORY FA	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumber 2. FA_FLAG = "Y"
10	REMOVE	RETURN ED TO	DEFECTIVE	REMOVE Transaction	Account alias Receipt from Float Org.	<b>Source:</b>

		DELL LOGISTI CS			Account Alias is "Field Service Recovery"	L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumb er
11	REMO VE	RETURN ED TO DELL LOGISTI CS	DISK ERASURE	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumb er
12	REMO VE	RETURN ED TO DELL LOGISTI CS	CUSTO MER FA	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumb er 2. FA_FLAG = "Y"
13	REMO VE	RETURN ED TO DELL	MANDAT ORY FA	REMOVE Transaction	Account alias Receipt from Float Org. Account Alias is	<b>Source:</b>

		LOGISTICS			"Field Service Recovery"	L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Serial Number Attributes:</b> 1. ReturnWaybillNumber 2. FA_FLAG = "Y"
14	DOA	RETURNED TO DELL LOGISTICS	DOA	INSTALL Transaction from Usable followed by REMOVE Transaction to Defective	Account alias issue from Float Org. Account Alias is "Field Service Usage" followed by Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b> For Transaction 1 - transaction_source_name = L_work_order  '- '  L_record_id  '- '  'INSTALL  <b>Reference</b> MAESTRO-PART_USAGE  For Transaction 2 - transaction_source_name = L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Reference</b> MAESTRO-PART_USAGE  <b>Serial Number Attributes:</b> 1. ReturnWaybillNumber for both the transactions
15	UNUSED	RETURNED TO DELL LOGISTICS	UNUSED	UNUSED Transaction	No Transaction	
16	UNUSED	LOST	UNUSED	LOST Transaction	Account Alias Issue with Account Alias as "Lost Part"	<b>Source:</b> transaction_source_name =



						L_work_order  '- '  L_record_id  '- '  'LOST'  <b>Reference</b> MAESTRO- PART_USAGE  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumb er
17	DOA	RETURN ED TO WAREH OUSE	DOA	INSTALL Transaction from Usable followed by REMOVE Transaction to Defective	Account alias issue from Float Org. Account Alias is "Field Service Usage" followed by Account alias Receipt from Float Org. Account Alias is "Field Service Recovery"	<b>Source:</b>  <b>For Transaction 1 -</b> L_work_order  '- '  L_record_id  '- '  'INSTALL  <b>Reference</b> MAESTRO- PART_USAGE  <b>For Transaction 2 -</b> L_work_order  '- '  L_record_id  '- '  'REMOVE'  <b>Serial Number</b> <b>Attributes:</b> 1. ReturnWaybillNumb er for both the transactions
18	DOA	LEFT ONSITE	PART RETENTI ON	Part Retention Issue	Account Alias Issue with Alias name "Site Adjustment"	<b>Source:</b> transaction_source _name = L_work_order  '- '  L_record_id  '- '  'PR_ISSUE'  <b>Reference</b> MAESTRO- PART_USAGE

19	UNUSED	LEFT ONSITE	RTR	Error: No transaction s needed for UNUSED - RTR	No Transaction	
20	UNUSED	RETURN ED TO WAREH OUSE	UNUSED	UNUSED Transaction	No Transaction	
21	UNUSED	LEFT ONSITE	PART RETENTI ON	Part Retention Issue	Account Alias Issue with Alias name "Site Adjustment"	<b>Source:</b> l_work_order  '- '  l_record_id  '- '  PR_ISSUE'  <b>Reference</b> MAESTRO- PART_USAGE
22	NO ACTIO N TAKEN			Error: No transaction s needed for NO ACTION TAKEN	No Transaction	
23	UNUSED	LEFT ONSITE FOR PICKUP	UNUSED	UNUSED Transaction	No Transaction	

---

## 2.5 Validation & Error Handling Requirements

The Break-fix Interface from Maestro to Oracle Cloud must include robust validation and error handling mechanisms to ensure data integrity and seamless operation. The following requirements outline the necessary validation checks and error handling procedures:

### 1. Data Validation:

- Quantity cannot be NULL, 0 or Negative
- DPS Number, Line Number, Part Number, Source Org, Source subinv, destination

location cannot be NULL

- Orgs should be active and destination location should be a valid float Org

### 2. Error Handling:

- Error Logging: All validation errors should be logged with detailed information, including the field name, error type, and a descriptive message. This log should be accessible for troubleshooting and audit purposes.

- Error Notification: Implement a notification system to alert relevant stakeholders (e.g., system administrators, data managers) of validation errors. Notifications can be sent via email or integrated messaging systems.

- Retry Mechanism: For transient errors (e.g., network issues, temporary unavailability of Oracle Cloud services), implement a retry mechanism with configurable retry intervals and maximum retry attempts.

### 3. Integration with Oracle Cloud:

- API Response Handling: Properly handle responses from Oracle Cloud APIs, including success, warning, and error messages. Ensure that the interface can interpret and act upon these responses appropriately.

- Data Consistency: Ensure that data remains consistent between Maestro and Oracle. Any discrepancies should be flagged and resolved promptly

- Transaction Management: Implement transaction management to ensure that data is either fully processed or rolled back in case of errors, maintaining data consistency between Maestro and Oracle Cloud.

---

### 3. Functional Unit Test Cases

S.No.	Test Case	Expected Results
1		
2		
3		
4		
5		
6		
7		

---

## 4. Open and Closed Issues

### 4.1 Open Issues

Issue Id	Description	Opened By	Responsible	Due Date

### 4.2 Closed Issues

Issue Id	Description	Resolution	Signoff	Closed Date

---

## 5. Appendix