IFS Aviation Maintenance

AeroBuy API Guide

ICAM 26.2



Document Revision: 26.2-1

Publication Date: November 02, 2026

Contents

1 Introduction....5
Maintenix overview....5
Message schema versions currently supported....7
About Maintenix APIs....7

2 AeroBuy API scenarios....9
RFQ placement scenario....9
RFQ acknowledgement scenario....10
Quote receipt scenario....10
Purchase order placement and update scenario....11
Purchase order acknowledgment scenario....12
Order exception scenario....13
Monitor and resolve order exceptions....15
Shipment advisories scenario....16
Invoice creation and updating scenario....17
Invoice updates....20
Manually map miscellaneous lines on AeroBuy-created invoices....21
Coordinate between invoice and order agents to validate invoices....22
Invoice acknowledgement scenario....22
AeroBuy rejection scenario....23

3 Enabling and configuring the AeroBuy API in Maintenix....24
Accessing the AeroBuy API component....24
Licensing and authentication....24
Add your customer number to Maintenix....24
Updating your properties files....25
Update pubsub.properties....25
Update servers.properties....25
Update pubsub-subscriptions.properties....26
Update mx-servers.properties....27
Activate the RFQ event in the database....27
Activate PO triggers in the database....28
Granting users permission to send or receive AeroBuy messages....28
The AeroBuy API user....29
Grant SPEC 2000 permissions to a user role....29

Assign the SPEC 2000 role to a specific user 29
Managing SPEC 2000 compliant components 30
Creating SPEC 2000 compliant parts, manufacturers and vendors 30
Configuring SPEC 2000 enabled vendors 30
Mark vendors as SPEC 2000 enabled 31
Enable SPEC 2000 commands for vendors 31
Enabling alerts for SPEC 2000 vendors 32
Testing and verifying your integrations with Maintenix 34
Optional configuration 36
Set refresh value for PO Details page order exception indicator 36
Change adapter XML validation values 36
Process long aircraft registration codes 36
Configure a custom integration user 37
Enforce SPEC 2000 compliance 37
Update definition of the outbound message wrapper 42
Troubleshooting your AeroBuy configuration 43

4 About PubSub 46

Subscribers 46
Registering subscribers 47
Subscriptions 48
Defining server properties 49
HTTP subscriptions 50
JMS subscriptions 51
PubSub property files and subscriptions 54
Variables 55
Lookups 56
Property Encryption 56
CyberArk 57
Publishers 58
Data model 60
The message queue 60
Clustered environments 61
Alerts 61
Guaranteed delivery and handling failures 62
Handling failures 63
JMS failures 64

HTTP failures..... 64
How to retry failed transactions..... 65
Dispatchers..... 65
PubSub configuration..... 66
Enabling and disabling PubSub..... 69
Troubleshooting PubSub..... 69

Appendix A Sample messages..... 71
SOAP envelope..... 71
ATA_SparesQuoteRequest.xml..... 71
ATA_SparesQuoteInterim.xml..... 72
ATA_SparesQuoteFinal.xml..... 72
ATA_SparesOrder.xml..... 73
ATA_SparesMessageAck.xml..... 73
ATA_SparesOrderExc.xml..... 74
ATA_SparesSupplierShipNotice.xml..... 74
ATA_SparesInvoice.xml..... 75
ATA_InvoiceMessageAck.xml..... 75

Appendix B Message element descriptions..... 76
ATA_SparesQuoteRequest..... 76
ATA_SparesQuoteInterim..... 80
ATA_SparesQuoteFinal..... 82
ATA_SparesOrder..... 95
ATA_SparesMessageAck..... 102
ATA_SparesOrderExc..... 104
ATA_SparesSupplierShipNotice..... 110
ATA_SparesInvoice..... 121
ATA_InvoiceMessageAck..... 132
AEX_INVALIDS2K..... 134

1

Introduction

Maintenix overview

To improve the speed and efficiency of purchasing materials and communicating with vendors, you can use the Maintenix AeroBuy API to integrate with the AeroXchange AeroBuy application.

The AeroBuy API is part of the AeroBuy Parts Purchasing feature which includes the API, a page in the Maintenix user interface for quickly resolving exceptions, and additional shipment, taxes, and other charges information on the PO Invoice Details page. This feature is available to licensed users.

The AeroBuy API uses SPEC 2000 messages to support the following scenarios:

- Sending RFQs to vendors.
- Receiving RFQ acknowledgements and quotes from vendors.
- Sending purchase orders, order updates, or order cancellations to vendors.
- Receiving purchase order acknowledgements from vendors.
- Receiving purchase order exceptions from vendors.
- Receiving shipment information for issued purchase orders from vendors.
- Receiving invoice information from vendors for issued purchase orders and creating or updating invoices in Maintenix.
- Sending acknowledgement of invoice receipt to vendors.
- Receiving notice from AeroBuy when a message sent from Maintenix is invalid and rejected.

Before you use the AeroBuy API, fulfill the following prerequisites and be aware of the current limitations:

Prerequisites

You require a SPEC 2000 Customer Code.

Vendors must be SPEC 2000 enabled. See *Mark vendors as SPEC 2000 enabled*.

You must use SPEC 2000 part numbers with SPEC 2000 compliant manufacturer codes in purchase orders.

For additional configuration requirements, see *Enabling and configuring the AeroBuy API in Maintenix*.

The AeroBuy Parts Purchasing integration follows the ATA Spec2000 Revision 2014 schema. Some fields that are mandatory in the 2014 schema may not be mandatory in AeroBuy because AeroXchange supports clients who are still using older revisions of the schema. The Maintenix AeroBuy Parts Purchasing feature supports newer versions, which means invoice messages from vendors using older versions may be missing some mandatory information. Before you start to use the Maintenix AeroBuy Parts Purchasing integration, contact your suppliers and inform them which information, mandatory or optional, you expect to receive from them. For more information, see *ATA_SparesInvoice*.

Limitations

If an error occurs when sending an RFQ message to AeroBuy, an alert is raised. Users must cancel the current RFQ then create and send a new RFQ.

Creating new inbound shipments for purchase orders is not supported. If a shipment advisory is received without an existing matching shipment in Maintenix, an alert is raised. Updates for inbound shipments for purchase orders are not supported.

Shipment advisories with partial quantities are processed and matched to a shipment in Maintenix. However, if a second shipment advisory message with the remaining quantity is received before a new shipment is created in Maintenix, the advisory message is ignored.

Each line in a Maintenix purchase order is a separate purchase order in AeroBuy. Therefore miscellaneous line items that are not associated with parts, are not sent as part of messages.

If you issue a purchase order to a vendor and then change the order number in Maintenix, when you re-issue the purchase order, the vendor does not receive the change. If the order number must change, you must cancel the initial order and create a new order with the desired order number.

If a vendor provides more than one cash discount term on an invoice, the integration only accepts the term which offers the highest discount. The remaining cash discount terms are ignored.

Discounts and credit memos are not supported.

Account Codes and PO invoice line mapping must be done manually for miscellaneous invoice lines that are created automatically by the integration. These fields will not be auto-populated.

Message schema versions currently supported

The Maintenix AeroBuy API uses Air Transport Association of America (ATA) messages to communicate information.

In the current Maintenix release, the following message.xsd schema versions are supported:

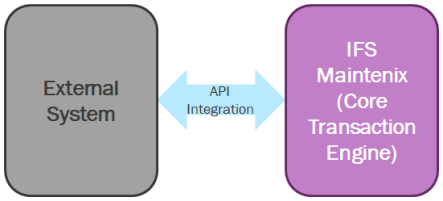
Message versions supported in the current Maintenix release

Message	Version
ATA_SparesQuoteRequest	R2014.1
ATA_SparesQuoteInterim	R2014.1
ATA_SparesQuoteFinal	R2014.1
ATA_SparesOrder	R2014.1
ATA_SparesMessageAck	R2007.1
ATA_SparesOrderExc	R2014.1
ATA_SparesSupplierShipNotice	R2014.1
ATA_SparesInvoice	R2014.1
ATA_InvoiceMessageAck	R2014.1
AEX_INVALIDS2K	AeroBuy Spec 2000 v16

About Maintenix APIs

Maintenix Application Programming Interfaces (APIs) support logical integrations with external business applications. You can also use Maintenix APIs for partial implementations, where your organization uses Maintenix in conjunction with an existing Enterprise Resource Planning (ERP) application, or for phased implementations, where workflows are supported by the existing ERP is transferred to Maintenix over time.

APIs simplify the task of integration, but do not deal with error handling between the integration layer of Maintenix and the external system. Messages are designed to describe the data in a flexible way, but message invocation is left open to customization. Timing considerations are particularly important when dealing with Web services that support a request-based integration as opposed to trigger-based integration.



Integration with Maintenix using APIs

There are three types of messaging scenarios available with the Maintenix APIs:

1. *Operational Service Messaging Scenario*: In this scenario, the external system sends an inbound message to Maintenix containing the information that Maintenix needs to function correctly. Upon successful processing of a message, Maintenix sends an outbound response message for confirmation.
2. *Querying Service Messaging Scenario:* In this scenario, the external system sends an inbound message to query information from Maintenix. Maintenix returns an outbound message containing the requested information.
3. *Notification Service Messaging Scenario*: In this scenario, certain Maintenix work flows create predefined events. These events are asynchonously pushed to an external system.

Note Not all APIs support all messaging scenarios.

2

AeroBuy API scenarios

RFQ placement scenario

When you issue a request for quote (RFQ) to vendors, Maintenix, sends an outbound ATA_SparesQuoteRequest.xsd (R2014.1) message to AeroBuy.

Summary

A purchasing agent prepares and sends an RFQ in Maintenix. An outbound RFQ message is sent to AeroBuy. The RFQ message contains all part lines in the Maintenix RFQ and is sent to each vendor. Once they are sent to AeroBuy, RFQs cannot be changed or deleted.

Pre-conditions

- The applicable vendor, manufacturer and parts are SPEC 2000 compliant. For more information, see *Managing SPEC 2000 compliant components*.

Messaging

1. Maintenix sends an outbound ATA_SparesQuoteRequest.xsd message.
2. Maintenix MAY receive an inbound ATA_SparesQuoteInterim.xsd message from the vendor via AeroBuy acknowledging the order.

Result

The quote is sent to the specified vendors.

If the message was not successfully published, an alert is raised. For more information about configuring alerts, see *Enabling alerts for SPEC 2000 vendors*.

RFQ acknowledgement scenario

When AeroBuy receives an ATA_SparesQuoteRequest.xsd (R2014.1) from Maintenix, the RFQ is available for vendors to view. Vendors can respond with an ATA_SparesQuoteInterim.xsd (R2014.1), but it is not required.

Summary

After the purchasing agent sends an RFQ from Maintenix, a vendor chooses to send an acknowledgement of the RFQ. The acknowledgement message does not contain any information that will update the quote in Maintenix, however, general comments may be included in the Vendor Remarks element.

Pre-conditions

- An RFQ is received and created in the AeroBuy portal.

Messaging

1. Maintenix receives an ATA_SparesQuoteInterim.xsd message.

Result

On the RFQ Details page:

- On the Quote Comparison tab, the Vendor Note is updated.
- A history note is added on the History tab

If the message was not successfully published, an alert is raised. For more information about configuring alerts, see *Enabling alerts for SPEC 2000 vendors*.

Quote receipt scenario

Vendors will send an ATA_SparesQuoteFinal.xsd message from AeroBuy to Maintenix with their quote information.

Summary

A vendor chooses to respond to an RFQ submitted by a purchasing agent.

Pre-conditions

- An RFQ is received and created in the AeroBuy portal.

Messaging

1. Maintenix receives an inbound ATA_SparesQuoteFinal.xsd message from AeroBuy communicating a vendor's response to an RFQ.

Result

On the RFQ Details page:

- On the Quote Comparison tab, RFQ details and vendor line attributes are updated including Quoted Part, Condition, Quote Qty, Unit of Measure, Unit Price, Currency Code, Lead Time, Effective To, Vendor Note, Quote Number, and Quote Date.
- A history note is added on the History tab

If the message was not successfully published, an alert is raised. For more information about configuring alerts, see *Enabling alerts for SPEC 2000 vendors*.

Purchase order placement and update scenario

When you issue purchase orders to vendors, Maintenix, sends an outbound ATA_SparesOrder.xsd (R2014.1) message to AeroBuy. This same message is sent when you update or cancel a purchase order, or to resend the original purchase order line details if you reject exception changes from vendors.

Summary

A purchasing agent prepares and issues an order in Maintenix. When the order is issued, an outbound order message is sent to AeroBuy. If you change any of the following fields and issue the order again, a message with your changes is sent:

- Vendor Code
- Priority
- Part Number
- Ship To
- Ship To Code
- Note to Vendor
- Price
- Quantity
- Promised By Date

A vendor can respond to acknowledge the receipt of the order or can request changes to the order. You can issue the order again with changes or cancel the order. Note: Each purchase order line in Maintenix is represented as its own purchase order in AeroBuy. If you change the order by adding more part lines or by canceling part lines this is the equivalent of creating a new purchase order or canceling a purchase order in AeroBuy.

Pre-conditions

When you create the purchase order, the part numbers and part manufacturers must be SPEC 2000 compliant.

The purchase order is ISSUED in Maintenix.

If the purchase order includes a line produced from a part request and has an aircraft associated with it, the aircraft registration code must be SPEC 2000 compliant.

Messaging

1. Maintenix sends an outbound ATA_SparesOrder.xsd message.
2. Maintenix receives either an inbound ATA_SparesMessageAck.xsd message from AeroBuy acknowledging the order, or an inbound ATA_SparesOrderExc.xsd message communicating vendor changes to the order.
3. If you accept changes, the purchase order is updated with the changes, but no message is sent. If you reject changes, Maintenix sends an outbound ATA_SparesOrder.xsd message containing the original purchase order line details. If the order exception is rejected, that means there were no changes to the purchase order and it remains in the ISSUED state. If you update the notes, then you must re-issue the order.
4. If you make changes to lines in a purchase order and reissue it, Maintenix sends an outbound ATA_SparesOrder.xsd message update for each line that you change.

Result

The order is sent to the vendor.

Each line in an Maintenix purchase order is a separate purchase order in AeroBuy. The separate POs are numbered with the original PO number appended with a hyphen and the line number. For example, issuing PO 8779 with three line items will result in three POs numbered 8779-1, 8779-2, 8779-3 in AeroBuy. Miscellaneous line items that are not associated with parts, are not sent as part of messages.

An alert is raised if the message was not successfully published. For more information about configuring alerts, see *Enabling alerts for SPEC 2000 vendors*.

Purchase order acknowledgment scenario

Depending on your contract with AeroBuy, Maintenix receives an ATA_SparesMessageAck.xsd (R2007.1) message acknowledging that AeroBuy and the vendor received your purchase order. AeroBuy can support either automatic vendor acknowledgement or manual vendor acknowledgement for when purchase orders are received.

Summary

You issue a purchase order and an acknowledgment is returned. The acknowledgement message does not explicitly mean that a vendor has reviewed the purchase order, only that the order was created in their system. When a purchase order has multiple line items, if one line is acknowledged, the status for the whole purchase order changes to acknowledged.

The purchase order status is updated on the Maintenix PO Details page.

Pre-conditions

Purchase order is received and created in the AeroBuy portal.

Messaging

1. Maintenix receives an ATA_SparesMessageAck.xsd message.

Result in Maintenix

On the **PO Details** page:

- The purchase **Order Status** changes to ACKNOWL (acknowledged). Because each line of a multi-line purchase order, is a separate purchase order in AeroBuy, if one line is acknowledged, the status of the purchase order is ACKNOWL.
- For a purchase order line, the **Price Type** value changes to CNFRMD (confirmed).
- For a purchase order line, the **Cnf** check box is selected to confirm the **Promised By Date**.

> **Note** If purchase order lines are added after the vendor's acknowledgement, the new lines do not inherit the confirmed status.

Order exception scenario

Maintenix receives ATA_SparesOrderExc.xsd (R2014.1) messages when vendors communicate changes to purchase orders such as a part number, scheduled date, unit of measure, unit price, or quantity.

Summary

After you issue a purchase order, vendors can change the purchase order and communicate the changes or exceptions to you. You view exceptions in Maintenix on the **Spec 2000 PO Exception Search** page and either accept or reject the changes. If you accept, no message is returned because both systems are now in alignment. If you don't accept or reject within a specified period of time (usually 24 hours), your acceptance is assumed and the vendor begins to process your order. If you reject the changes, a message containing the original details of the purchase order line is sent. This is required to revert the vendor's system back into alignment with Maintenix.

Pre-conditions

The vendor changes the purchase order in their system.

Note: A purchase order does not have to be in the ACKNOWL state in order for you to receive Order Exceptions. Not all vendors use acknowledgment messages, so it is possible to receive an order exception without first receiving an order acknowledgment message. Acknowledgment messages do not impact how order exceptions are handled.

Messaging

1. Maintenix receives ATA_SparesOrderExc.xsd messages communicating vendor changes.
2. You accept the changes: Maintenix does not send an outbound message.

3. You reject the changes: Maintenix sends an outbound ATA_SparesOrder.xsd message with the original purchase order. You do not need to issue the PO again unless you make additional changes to an order line.

Result in Maintenix

On the **PO Details** page:

- In the Order Information area, an order exception counter shows the number of unresolved exceptions associated with the PO. The order exceptions can be accessed by clicking on the count number. Order exceptions can also be accessed through the **Spec2000 PO Exception Search** menu item.
- On the History tab, a history note is added.
- Once an exception is raised, you must accept or reject it. You cannot edit the PO Details page until you resolve all outstanding exceptions. Quantity changes result in separate exceptions from other changes in an order line. For example if there is a change to the quantity and to the price for the same order line, there are two exceptions (one for quantity, another for price).
- After all pending exceptions are resolved, you can edit the purchase order.
- After an order exception is accepted, the purchase order is updated. The order remains in AUTH status, so if there was a price change that requires authorization above your approval level, you need to request authorization. After accepting an exception, you must issue the PO again.

On the **Order Exceptions** tab:

- If a vendor proposes an alternate part number for a line, the alternate part is marked as with one of three symbols:

Symbol	Indicates	User Action
	The Part Number does not exist in Maintenix.	Reject the exception. (The Accept button is disabled).
	The Part Number exists but does not belong to the same part group as the original part.	You cannot accept initially, but if the part is a valid alternate, you can add it to the part group and then accept.
	The Part Number exists and is listed in the same Part Group as the original part.	You can accept the exception.

- If a vendor proposes a quantity, price, or promised by date change, the proposed change is displayed in bold.

Monitor and resolve order exceptions

If your organization uses the AeroBuy Parts Purchasing feature, vendors communicate purchase order changes (called exceptions) which you monitor and resolve on the Spec 2000 PO Exception Search page.

About this task

The PO Details page displays an exception counter and you cannot edit orders until all order exceptions are accepted or rejected. If you don't accept or reject within a specified period of time (usually 24 hours), your acceptance is assumed and the vendor begins to process your order.

Proposed changes from vendors are displayed in bold font on the **Spec 2000 PO Exception Search** page along with remarks if vendors provide them.

If you accept an exception, the purchase order is updated in Maintenix. On the PO Details page, the PO line is updated. The order remains in AUTH status, so if there was a price change that requires authorization above your approval level, you need to request authorization.

When all order exceptions are dealt with, the exception counter disappears and the order is editable again.

To find exceptions that you're responsible for, you can filter your view of the page by selecting your name from the Buyers list or by selecting a specific vendor, purchase type, or purchase order. To find exceptions for orders that were automatically created as a result of a stock low action and that were auto-issued through AeroBuy, you can filter on purchase type combined with selecting MXPURCHASE from the Buyers list filter.

Steps

1. To go to the **Spec 2000 PO Exception Search** page:
   - On your role menu, click **Spec 2000 PO Exception Search** or,
   - On the **PO Details** page, **Order Information** area, **Order Exceptions (s) pending** field, click the red number (exception counter).
2. On the **Spec 2000 PO Exception Search** page, **Order Exceptions** tab, for each exception, click **Accept** or **Reject**.

   If a vendor proposes an alternate part number for a line, the alternate part is marked with one of three symbols:

Symbol	Indicates	User Action
	The Part Number does not exist in Maintenix.	Reject the exception. (The Accept button is disabled).
	The Part Number exists but does not belong to the same part group as the original part.	You cannot accept initially, but if the part is a valid alternate, you can add it to the part group and then accept.

Symbol	Indicates	User Action
	The Part Number exists and is listed in the same Part Group as the original part.	You can accept the exception.

3. In the **Confirmation Required** window, click **OK**.

   In AeroBuy, each line of an Maintenix PO is a separate PO, so the confirmation cites the Maintenix PO number plus a line number (for example: PO103-1, PO103-2, PO103-3).

4. If you accepted an order exception, authorize if required, and issue the PO.

The PO is the same now in Maintenix and in the vendor's system -- unless you did not accept or reject an exception within the specified period of time. In this case an exception is still displayed on the PO Details page while the vendor may already be processing your order. If you intended to accept the order, then accept the exception in Maintenix so that both systems are aligned. If you intended to reject the order, contact the vendor to resolve the discrepancy.

Shipment advisories scenario

Maintenix receives an inbound ATA_ SparesSupplierShipNotice.xsd (R2014.1) message from AeroBuy when vendors communicate information about shipments.

Summary

After a purchase order is processed, vendors ship the part and communicate that the part was shipped. The vendor resends the message if shipment details become available or change.

Pre-conditions

An inbound shipment is created in Maintenix that corresponds directly to the purchase order.

A vendor has shipped a part to fulfill an order that was placed.

Messaging

1. Maintenix receives an inbound ATA_ SparesSupplierShipNotice.xsd message from AeroBuy communicating that a part was shipped.

Result in Maintenix

The shipment status updates from PENDING to IN-TRANS.

On the Shipping Details page, the name of the shipping carrier and the waybill number are populated (if provided by the vendor).

A history note is added to the shipment. The note contains additional information (not found on the shipment details page) including weight, package dimensions, transport method and payment type.

If a shipment advisory is received without an existing matching shipment in Maintenix, a note is recorded and an alert is raised.

If there is more than one shipment against the purchase order with the provided waybill number, a No Match For Shipment Advisory Alert is raised.

If there is more than one shipment against the purchase order with an empty waybill number, a No Match For Shipment Advisory Alert is raised.

For more information about alerts, see *Enabling alerts for SPEC 2000 vendors*.

Invoice creation and updating scenario

Maintenix receives an inbound ATA_SparesInvoice message when vendors request payment. A successfully received message creates an invoice in Maintenix. The same message is received if vendors update invoices.

Invoices must be received to three-way match and close purchase orders. Invoice messages must always include the part number, price, and quantity information that is required for three-way matching of the invoice with the order and with the materials received. It is important to note that if an Maintenix PO has multiple lines, in AeroBuy there will be a PO and a corresponding invoice for each line. This behavior means that a Maintenix PO can have multiple invoices in varying states. An invoice that is validated for payment only closes an Maintenix PO if it's the last invoice validated for payment for the whole order.

Vendors reuse invoice numbers so invoices with the same invoice number and vendor code can be automatically created in Maintenix if the existing invoices have a PAID or CANCEL status. If more than one invoice in the system has the same vendor code and invoice number, only one can have a TOBEPAID status.

Summary

After purchase orders are issued at least once, vendors can send invoices which are received and created in Maintenix. There is a separate invoice for each part line in an Maintenix purchase order because each line is a separate purchase order in AeroBuy. Invoices are automatically mapped to purchase order lines. The accounts payable accountant assesses the invoice in Maintenix, reviews it, and clicks Validate Payment. If there is an error on the invoice, it cannot be edited in Maintenix and the accountant must contact the vendor directly and request that the vendor send an updated invoice message. Vendors can send updates to existing invoices or can create new invoices that may supersede existing invoices. For more information, see *Invoice updates*.

Preconditions

The vendor publishes the purchase order invoice from their system of choice.

A corresponding purchase order (and line in a purchase order) exists in Maintenix.

The purchase order was issued at least once and has a status of OPEN, AUTH, ISSUED, ACKNOWLEDGED, PARTIAL, or RECEIVED.

Part numbers and SPL codes in the invoice message correspond to the Maintenix purchase order line.

Messaging

1. Maintenix receives an ATA_SparesInvoice message communicating vendor prices, applicable taxes, charges and cash discounts.
2. Maintenix sends an ATA_InvoiceMessageAck.xsd message to the vendor acknowledging receipt of the invoice.
3. Maintenix receives an ATA_SparesInvoice message communicating an update to an existing invoice using the same invoice number.
4. Maintenix sends an ATA_InvoiceMessageAck.xsd message to the vendor acknowledging receipt of the invoice.
5. Maintenix receives an ATA_SparesInvoice message communicating that a referenced, existing invoice is superseded by a new invoice with a new number. The message contains all the information about the new invoice.
6. Maintenix sends an ATA_InvoiceMessageAck.xsd message to the vendor acknowledging receipt of the new invoice.

Results in Maintenix

A PO Invoice is created or updated in Maintenix. You can access invoices in different ways:

- From your Accounts Payable Accountants To Do List page, My Open PO Invoices page.
- Do a PO Invoice Search.
- From the PO Details page, Receipt & Returns tab.

On the PO Details page, Receipt and Returns tab:

- The PO Invoices section displays all invoices applicable to the PO, including invoices that were created by a user in Maintenix. Click the invoice number to open the Invoice Details page. You can review the invoice and validate it for payment.

On the To Do List page, My Open Invoices tab:

- On this tab, you see the invoices for which you are the Order Invoice Contact. Click the invoice number to open Invoice Details page.

On the PO Invoice Details page:

- Numerous fields are populated. The invoice line displays the part, quantity, price, tax sales point date, and vendor remarks. The PO Invoice links to a PO. If provided by the vendor, shipment related details, taxes & other charges will be displayed on the Details tab. This information is all specific to the PO invoice line.
- Some buttons and fields (that are visible or enabled on the pages of invoices that are created manually) are hidden or disabled when invoices are created or updated through the AeroBuy API. This restriction ensures that information in Maintenix and the vendor systems is aligned.

- Miscellaneous invoice lines are automatically created if taxes or charges are provided by the vendor. There is one invoice line for the Total Tax Amount, and one invoice line for the Total Other Charge Amount. Before you can map the miscellaneous lines to a corresponding miscellaneous lines on a purchase order, you must manually update the miscellaneous line's charge account. See *Manually map miscellaneous lines on AeroBuy-created invoices*.
- If a vendor provides more than one cash discount term, only the term with the highest discount is displayed in Maintenix. If provided, sales taxes, value added taxes and other charges are displayed in an itemized list.
- On the History tab, a history note is added any time an invoice is created, updated, or superseded by the vendor.
- Invoices can only be validated for payment when the part number, price, and quantity match in the purchase order line, the invoice line, and the shipment line.
- Invoices cannot be updated by vendors after an invoice is validated for payment and is in the TOBEPAID or PAID state, or after an invoice is canceled and is in the CANCELED state.
- Validating an invoice for payment closes an Maintenix purchase order if it covers the remaining quantity received which has not yet been validated for payment.

Alerts are raised when an inbound ATA_SparesInvoice message cannot be processed for the following reasons:

- A purchase order that corresponds to the invoice message does not exist in Maintenix.
- A purchase order line that corresponds to the invoice message does not exist on a specific purchase order in Maintenix.
- The purchase order is historical because it was closed or canceled.
- The invoice and purchase order information doesn't match for the following:
  - CIC - Customer Code
  - SPL - Vendor Code
  - PNR - Part Number
  - UNP.ICR - Currency Code
- The invoice date is not provided by the vendor.
- An invoice with the same invoice number and vendor code already exists. The message is not an update (no ADS number) and the existing invoice is not cancelled or paid.
- The ADS number is invalid - the ADS number cannot be the same, lower than, or two or more numbers greater than the previous ADS number.

Invoice updates

AeroBuy API-created invoices cannot be edited in Maintenix, so if there is an error in the invoice, the accountant must contact the vendor outside of Maintenix and request that the vendor send an updated invoice using the ATA_SparesInvoice message.

Vendors can send updates to existing invoices or create new invoices that may supersede existing invoices. All updates are recorded on the History tab.

Updates that use the same invoice number

The update includes the ADS element. The first update to an existing invoice must have an ADS value of 1. For subsequent updates to the same invoice, increment the ADS number sequentially.

New invoice with a different invoice number

The vendor creates a new invoice with a different invoice number and provides a reference to the original invoice number. Maintenix tries to overwrite the referenced invoice; if it cannot, a new invoice is created. If invoices are superseded, accountants may cancel the original invoice in Maintenix. An invoice will only be superseded and overwritten by another inbound ATA_SparesInvoice message if the following criteria are met:

- The new invoice has the same purchase order number and line number (CPO) as another invoice in Maintenix.
- The new invoice has a unique invoice number (INV) that does not already exist in Maintenix for the same vendor.
- The new invoice does not have an adjustment sequence number (ADS).
- The new invoice has a DEBIT invoice type code (ITC=D).
- The new invoice has a reference invoice number (RIN) that is the same as the existing invoice and the CPOs of the new and the superseded invoices match.
- The new invoice has the same or greater Invoice Quantity (INQ) as the purchase order line quantity. If not for this, the invoice could simply be another invoice for a remaining quantity (not yet invoiced).

An invoice can be updated by either a debit invoice or a credit invoice. The Invoice Type Code (ITC) is either C for credit or D for debit. If an invoice is updated by a credit invoice, the quantity and unit price cannot result in a negative value. If the quantity or unit price in the credit invoice is greater than the quantity or unit price specified on the invoice line, an alert is raised.

A vendor may also decide to create a new invoice rather than update an existing one. In this case, the vendor will not use the same invoice number, nor will they reference an existing invoice with an RIN. If a new invoice is created in place of an original invoice, the accountant should cancel the original invoice in Maintenix.

Manually map miscellaneous lines on AeroBuy-created invoices

Invoices that are created through the Maintenix AeroBuy API are automatically mapped to purchase order lines, but if the invoice contains miscellaneous lines, you must update the Account, and manually map the lines before you can validate the invoice for payment.

About this task

See also *Coordinate between invoice and order agents to validate invoices*.

Miscellaneous invoice lines are automatically created if taxes or charges are provided by the vendor and they are created with an Account of VOID. There is one invoice line for the Total Tax Amount, and one invoice line for the Total Other Charge Amount. The Total Tax Amount line displays the sum of all the Sales and Value Added Taxes provided by the vendor for a given line. The Total Other Charge Amount displays the sum of all other charges provided by the vendor for a line. You can see an itemized breakdown of taxes and charges on the Invoice Details tab.

Note that while you can edit the account, most fields on AeroBuy API-created invoices are not editable (and most buttons are disabled or hidden). If there are errors on the invoice, contact the vendor and request an updated invoice. For more information, see the *Maintenix AeroBuy API Guide.*

Steps

1. To find an invoice, do one of the following:
   - On your **To Do List**, **My Open Invoices** tab, click the link for an invoice.
   - From your role menu, click **PO Invoice Search**, click **Search**, and then click the link for an invoice.
2. On the **PO Invoice Details** page, **PO Invoice Lines** tab, click **Edit PO Invoice Lines**.
3. On the **Edit PO Invoice Lines** page, in the **Account** column for the line, enter an account and click **OK**.
4. Match each miscellaneous line.

   If there's no matching order line, go to the order, add the miscellaneous line to the order, and assign an account to the line. Return to the invoice, and click **Map PO Invoice Line to PO Line** again. Order lines only appear in the mapping search if they have the same **Charge Account** as the **Invoice** line you are trying to map. The name of the line is irrelevant when it comes to mapping invoice lines with order lines.

   When all the lines are matched, you can validate the invoice for payment. If the invoice and the PO don't match and there are remaining errors, you can edit the PO or contact the vendor to request an update to the invoice.

   If you mapped the miscellaneous lines, but then the vendor sends an invoice update, the miscellaneous line accounts are reset to VOID and must be updated and mapped again.

Coordinate between invoice and order agents to validate invoices

If there are matching errors when you attempt to validate an AeroBuy-created invoice for payment accounts payable accountants may need to coordinate with purchasing agents to resolve the issues. In Maintenix, you can pass invoices back and forth for action by the appropriate person.

About this task

In some organizations accounts payable agents may contact vendors when invoices don't match purchase orders, while in other organizations, purchasing agents contact vendors to resolve discrepancies.

To assign the invoice to a purchasing agent or back to an accounts payable accountant, use the following steps:

Steps

1. On the **PO Invoice Details** page, **Details** tab, **Details** area, click .
2. On the **Edit Invoice** page, in the **Order Invoice Contact** field, enter the name of the person you want to assign to the invoice.
3. Click **OK**.
   The invoice is listed on the **My Open Invoices** tab of the assigned user. When the discrepancy is resolved, the assigned user re-assigns the invoice to the appropriate person to 3-way match the invoice and close the purchase order.

Invoice acknowledgement scenario

Maintenix sends an outbound ATA_InvoiceMessageAck.xsd (R2007.1) message to AeroBuy communicating to the vendor that an invoice was received. The acknowledgement is sent when new invoices are received and when existing invoices are updated.

Summary

When a vendor sends an invoice and the invoice is successfully created in Maintenix, an acknowledgement message is automatically sent to AeroBuy. In the vendor system, the invoice status is updated to acknowledged.

Preconditions

A valid invoice was received and created in Maintenix.

Messaging

1. Maintenix receives an ATA_SparesInvoice message communicating the vendor unit price, applicable taxes, charges and cash discounts.
2. Maintenix sends an ATA_InvoiceMessageAck.xsd message to AeroBuy acknowledging receipt of the invoice.
3. Maintenix receives an ATA_SparesInvoice message communicating an update to an existing invoice using the same invoice number.

4. Maintenix sends an ATA_InvoiceMessageAck.xsd message to AeroBuy acknowledging receipt of the invoice.
5. Maintenix receives an ATA_SparesInvoice message communicating that a referenced, existing invoice is superseded by a new invoice with a new number.
6. Maintenix sends an ATA_InvoiceMessageAck.xsd message to the vendor acknowledging receipt of the invoice.

Result

There are no updates made in Maintenix, but in AeroBuy and the vendor system, the invoice status changes to acknowledged.

AeroBuy rejection scenario

Maintenix receives an inbound AEX_INVALIDS2K.xsd message from AeroBuy if a message that Maintenix sent was invalid (in format or construction). Receipt of this message raises an alert.

Summary

If Maintenix sends an invalid message, AeroBuy communicates this event to Maintenix.

Pre-conditions

An invalid message is sent from Maintenix.

AeroBuy must have AEX_INVALIDS2K set up as a message. In addition, it can optionally be set up as an email notification.

Messaging

Maintenix receives an inbound AEX_INVALIDS2K.xsd message from AeroBuy.

Result

An alert is raised which provides the reason for the error. For more information about alerts, see *Enabling alerts for SPEC 2000 vendors*.

3

Enabling and configuring the AeroBuy API in Maintenix

Before using the AeroBuy Parts Purchasing API, Maintenix must be specifically configured to interact with it.

Accessing the AeroBuy API component

The AeroBuy component is available at the following endpoint:

`https://<HOST_NAME>:<PORT>/aerobuy/ws/spec2000/2014`

Licensing and authentication

To use this feature, your organization must be licensed for the AeroBuy Parts Purchasing feature. If your current license does not include this feature, contact your IFS representative.

You will also need to have an account with AeroXchange, who own the AeroBuy product with which this API integrates. Your agreement with them will determine your authentication method, which may involve basic authentication or digital certificates.

Establishing the connectivity related to certificates and 2-way SSL configuration is outside of the scope of this guide.

Add your customer number to Maintenix

You receive your customer number when you create an account with AeroXchange.

Steps

In the REF_SPEC2K_CUST table, enter your customer number.

Follow SPEC 2000 rules, using three to five characters.

Updating your properties files

Properties files are used to store configurable parameters, in an easily accessible way.

The AeroBuy API configuration properties files are stored in the `/mxi/apps/assetmanagement/resource/property/` folder of your Maintenix installation.

Depending on how Maintenix is installed on specific Network Interface Controllers, you may need to provide a specific IP address in your endpoints, instead of using the default value of `localhost`. Or, if the AeroBuy API and Maintenix are installed at different locations, you must change the host and port values to describe the location of your AeroBuy API installation.

Important You must restart your server for changes to properties files to take effect.

Update pubsub.properties

This file enables the internal Maintenix publish-subscription integration framework called PubSub.

Steps

1. Navigate to the `/mxi/apps/assetmanagement/resource/property/` folder where your instance of Maintenix is installed.
2. Open the `pubsub.properties` file in a text editor.
3. Set the `com.mxi.pubsub.config` parameter to `true`.

```
com.mxi.pubsub.config.enabled=true
```

4. Save and close the file.

Important You must restart your server for changes to properties files to take effect.

Update servers.properties

This file contains variables that are used by the `pubsub-subscriptions.properties` file.

About this task

You may choose to hardcode the URL and credential information in your `pubsub-subscriptions.properties` file. In this case, you do not need to update your `servers.properties` file.

Steps

1. Navigate to the `/mxi/apps/assetmanagement/resource/property/` folder where your instance of Maintenix is installed.
2. Open the `servers.properties` file in a text editor.
3. Uncomment the lines under `Aerobuy server example` that corresponds to the type of authentication your connection will use.
   - If using basic authentication (Aerobuy Basic Auth subscription), the `aeroxchange.aerobuy.username` and `aeroxchange.aerobuy.password` properties are used.
   - If using digital certification (Aerobuy Digital Certs subscription), the `aeroxchange.aerobuy.username` and `aeroxchange.aerobuy.password` properties are not required.
4. Save and close the file.

> **Important** You must restart your server for changes to properties files to take effect

Update pubsub-subscriptions.properties

This file contains the subscriptions for all channels, including AeroBuy.

Prerequisite

- You have arranged with AeroXchange to use either basic authentication or digital certification for authentication.

About this task

To effectively use AeroBuy functionality in Maintenix, you must subscribe to the following AeroBuy channels:

- MX_ATA_SPARES_ORDER_MESSAGE
- MX_AEROBUY_ATA_SPARES_INVOICE_ACK_MESSAGE
- MX_ATA_SPARES_QUOTE_MESSAGE

Steps

1. Navigate to the `/mxi/apps/assetmanagement/resource/property/` folder where your instance of Maintenix is installed.
2. Open the `pubsub-subscriptions.properties` file in a text editor.
3. Uncomment the code block that corresponds to the authentication model your organization has arranged to use with AeroXchange. It will be one of the following:
   - `Aerobuy Basic Auth subscription`, OR
   - `Aerobuy Digital Certs subscription`

4. (Optional) You may choose to hardcode any of: `username`, `password` or `url` for either of the subscriptions, rather than using the variables.

   The variables used in this properties file are defined in `servers.properties`. For more information, see *Update servers.properties*.
5. Save and close the file.

> **Important** You must restart your server for changes to properties files to take effect.

Update mx-servers.properties

This file points the AeroBuy extension to the MX API endpoint.

About this task

Update this file with the appropriate values for your Maintenix installation.

Steps

1. Navigate to the `/mxi/apps/assetmanagement/resource/property/` folder where your instance of Maintenix is installed.
2. Open the `mx-servers.properties` file in a text editor.
3. Edit `maintenix.url=http://<localhost>:<port>` with the host and port values of your Maintenix installation location.
4. Save and close the file.

> **Important** You must restart your server for changes to the properties files to take effect.

Activate the RFQ event in the database

Run an SQL statement to ensure that the MX_RFQ_UPDATED event is enabled.

Steps

In your Maintenix database, run the following statements:

```
-- Ensure the the MX_RFQ_UPDATED event is enabled
UPDATE int_event_config
   SET enabled_bool = 1,
    after_snapshot_bool = 1
 WHERE event_type_cd = 'MX_RFQ_UPDATED'
   ;
commit;
```

Activate PO triggers in the database

Run an SQL statement to activate the issuing and canceling requests for purchase orders.

Steps

In your Maintenix database, run the following statements:

```
-- Activate the trigger on issuing a purchase order
UPDATE utl_trigger
   SET active_bool = 1
 WHERE trigger_cd='MX_PO_ISSUE'
   ;

-- Activate the trigger on cancelling a purchase order
UPDATE utl_trigger
   SET active_bool = 1
 WHERE trigger_cd = 'MX_PO_CANCEL'
   ;

commit;
```

Granting users permission to send or receive AeroBuy messages

To facilitate the Maintenix-AeroBuy integration, a default mxspec2000pointegration user has the permissions required to authenticate and authorize inbound from AeroBuy to Maintenix. You can choose to grant these permissions to a specific user or user group, as well as select users or user groups to send outbound messages from Maintenix to AeroBuy.

There are two permission sets in Maintenix related to AeroBuy messages:

- Receive Spec 2000 Message
  - Contains all the permissions necessary to receive Spec 2000 messages.
  - Must be assigned to the user that AeroBuy sends its message to.

    Note This user is configured in AeroBuy, not on your Maintenix server.

- Send Spec 2000 Message
  - Contains all the permissions necessary to send Spec 2000 messages.
  - Required by any role that issues purchase orders, handles order exceptions or sends RFQs (i.e. triggers outbound AeroBuy messages).

By default, the SPEC2K user role is granted the Receive Spec 2000 Message and the Send Spec 2000 Message permission sets.

The AeroBuy API user

To facilitate the Maintenix-AeroBuy integration, a default user called mxspec2000pointegration is created with the SPEC2K role, which gives it the permissions required to authenticate and authorize inbound and outbound messages from AeroBuy to Maintenix.

The mxspec2000pointegration user has the following details:

- First Name: mxspec2000pointegration
- Last Name: mxspec2000pointegration
- Username: mxspec2000pointegration
- HR Code: SYSTEM
- Roles assigned: SPEC2K

Grant SPEC 2000 permissions to a user role

Give all users with a specific role permission to send AeroBuy messages, receive AeroBuy messages or both.

Steps

1. Click the down arrow next to the Maintenix logo.
2. Select **Administrator**, then **Role Search**.
3. Search for a role, then select it from the list.
4. Click the **Permission Sets** tab.
5. Under the **Spec 2000** setting, choose one or both of the following permission sets:
   - Receive Spec 2000 Message
   - Send Spec 2000 Message
6. Click **OK**.

Assign the SPEC 2000 role to a specific user

Give a specific user permission to send and receive AeroBuy messages.

Steps

1. Click the down arrow next to the Maintenix logo.
2. Select **Administrator**, then **Role Search**.
3. Search for **Role Code** SPEC2K, then select it from the list.
4. On the **Users** tab, click **Assign User**.
5. Search for a user, then select the user from the list.
6. Click **Assign User**.

Managing SPEC 2000 compliant components

The AeroBuy API requires the part, manufacturer and vendor information sent in every message to be compliant with SPEC 2000 specifications.

Creating SPEC 2000 compliant parts, manufacturers and vendors

All parts and manufacturers you intend to use with the AeroBuy API must adhere to SPEC 2000 requirements. When creating parts and manufactures in Maintenix, make sure your part numbers and manufacturer codes meet the following requirements.

SPEC 2000 part number requirements

- Only alphanumerical characters and dashes
  - *Exception:* no lowercase or uppercase letter "O"
- Dashes can only be used to separate numbers
- The last character of a part number cannot be a dash
- Maximum of 32 characters

SPEC 2000 manufacturer code requirements

- Only alphanumerical characters
- Must be exactly 5 characters

SPEC 2000 vendor code requirements

In addition to the following code requirements, vendors must also be SPEC 2000 enabled. For more information, see *Configuring SPEC 2000 enabled vendors*.

- Must be exactly 5 characters

Configuring SPEC 2000 enabled vendors

To use the AeroBuy API to communicate with vendors, vendors must be configured in Maintenix to allow for communication in accordance with the SPEC 2000 specification.

Vendors must be created and configured as per the *Maintenix User Guide*, in addition to meeting the SPEC 2000 requirements.

Mark vendors as SPEC 2000 enabled

Before Maintenix can communicate with vendors using SPEC 2000 standard messages, you must mark vendors as SPEC 2000 enabled.

About this task

Steps

1. Navigate to the **Vendor Details** page, using the **Vendor Search** tool on your role menu.
2. On the **Vendor Details** page, **Details** tab, **Details** area, click **Edit Vendor**.
3. Select the **Spec 2000 Enabled** check box.
4. Click **OK**.

Enable SPEC 2000 commands for vendors

Before Maintenix can communicate with vendors using the AeroBuy Parts Purchasing API, you must enable SPEC 2000 commands for each vendor.

Steps

1. Navigate to the **Vendor Details** page, using the **Vendor Search** tool on your role menu.
2. On the **Communications** tab, select the following **Spec 2000 Commands**.

Configuration Item	Description
S1QUOTES	Allows Maintenix to send RFQs to and receive RFQ responses from SPEC2000 vendors automatically
S1BOOKED	Allows Maintenix to send POs to SPEC2000 vendors automatically when orders are issued.
S1ORDEXC	Allows vendors to send requests to change POs in Maintenix. This command should be enabled if S1BOOKED is enabled.
S1SHIPPED	Allows vendors to send shipment advisories to Maintenix for POs that were sent using the S1BOOKED command. This command should be enabled if S1BOOKED is enabled.
S1INVCE	Allows vendors to create and update invoices in Maintenix. This command is often enabled if S1BOOKED is enabled.

3. Click **OK**.

Enabling alerts for SPEC 2000 vendors

Before a SPEC 2000 AeroBuy vendor is enabled, the following alerts should be configured for roles that are assigned to users that interact with AeroBuy registered vendors.

You can enable alerts in Maintenix on the **Alert Setup** page. Below is a list of alerts that are specific to AeroBuy implementations with suggested roles that may want to receive them.

Alerts that support the AeroBuy Parts Purchasing API

Alert Name	Category	Description	Alerted Roles *(examples only)*
Aeroexchange Rejection Message	PO	An Aeroexchange Rejection message was received.	*Purchasing Manager*
ATA_SparesOrder Message Failed To Be Published	PO	Maintenix failed to publish the ATA_SparesOrder message due to an unexpected error.	*Purchasing Manager*
Inbound Spec 2000 Message Failed	PO	Maintenix failed to process an inbound SPEC 2000 message.	*Purchasing Manager*
Purchase Order Exception	PO	An exception was raised against a purchase order.	*Purchasing Manager*
Spec 2000 Message Fails Schema Validation	PO	A SPEC 2000 inbound message failed the schema validation.	*Purchasing Manager*
No Matching Shipment Record for Shipment Advisory	SHIPMENT	No shipment record exists in Maintenix for a shipment advisory received from an external system.	*Purchasing Agent*
Shipment Advisory	SHIPMENT	A shipment advisory was received from an external system.	*Purchasing Agent*

Alert Name	Category	Description	Alerted Roles *(examples only)*
AeroBuy invoice not created or updated	INVOICE	An inbound AeroBuy ATA_SparesInvoice message was not processed because of an error.	*Accounts Payable Accountant*
AeroBuy invoice not created or updated - internal error	INVOICE	An inbound AeroBuy ATA_SparesInvoice message was not processed because of an internal error.	*Accounts Payable Accountant*
AeroBuy invoice created or updated with warnings	INVOICE	An inbound AeroBuy ATA_SparesInvoice message was processed, but there are warnings.	*Accounts Payable Accountant*
Aerobuy message could not be received.	Aerobuy	An inbound Aerobuy message could not be received.	
Aerobuy message was not delivered.	Aerobuy	An outbound AeroBuy message was not delivered.	
AeroBuy RFQ updated with warnings	RFQ	An inbound AeroBuy ATA Spares Quote message was processed, but there were warnings.	*Purchasing Manager*
RFQ Message Failed To Be Published	RFQ	Maintenix failed to publish the RFQ message due to an unexpected error.	*Purchasing Manager*

Alert Name	Category	Description	Alerted Roles *(examples only)*
AeroBuy RFQ not updated	RFQ	An inbound AeroBuy ATA Spares Quote message was not processed because of an error.	*Purchasing Manager*

Testing and verifying your integrations with Maintenix

You can send test messages to verify your integration using a SOAP application such as soapUI , which has an open source version available. You can download soapUI from *http://sourceforge.net/projects/soapui/files/*.

The following steps outline the typical actions that you must perform in soapUI or other comparable applications. For specific instructions related to your chosen SOAP application, consult that product's documentation.

1. Obtain the WSDL (Web Services Description Language) request definition.

   Before you can send a message, you must obtain the WSDL request definition file for Maintenix APIs. You can find a copy of the WSDL file (`Request.wsdl`) in your installation package, or at the following URL:

   ```
   http://
   <username>:<password>@<application_server_hostname>:<admin_http_port>/
   mxcorews/services/RequestWS?wsdl
   ```
2. In your SOAP application, create a new project. When asked for the WSDL location, specify the location where you stored the `Request.wsdl` file.
3. Enter the URL that corresponds to the API that you are invoking.
4. Typically, the SOAP application will generate a blank request message. You can modify that message, or replace it with the contents of one of the sample messages that are included with the Maintenix API documentation. To send an echo message, copy the text from the *sample echo message* that follows.
5. In order to connect to Maintenix, you must provide your authentication credentials, including your Maintenix username and password.

> **Note** If you are using soapUI, you must also enable the option to add authentication information to outgoing requests. You can find this option in your soapUI preferences.

6. In your SOAP application, send your message. If your message is successful, you will receive a successful response message from the CTE.

The following is a sample echo message that you can send to the `http://<server_name>:63081/mxi/echo/1.0`

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header>
        <head:mode
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:head="http://xml.mxi.com/xsd/integration/header/1.1"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">sync</
head:mode>
    </soapenv:Header>
    <soapenv:Body>
        <req:request
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:req="http://xml.mxi.com/xsd/integration/
request/1.1">
            <echo xmlns="http://xml.mxi.com/xsd/test">Hello, World</
echo>
        </req:request>
    </soapenv:Body>
</soapenv:Envelope>
```

Endpoint URL:The following is a sample response from a successful inbound echo message:

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <resp:response msg_id="1dcf6b23-13c0-4130-afdc-9975110aa7a7"
 msg_status="COMPLETE"
            xmlns:resp="http://xml.mxi.com/xsd/integration/
response/1.1">
            <echoResponse xmlns="http://xml.mxi.com/xsd/test">Hello,
 World</echoResponse>
        </resp:response>
    </soap:Body>
</soap:Envelope>
```

Optional configuration

Set refresh value for PO Details page order exception indicator

On the PO Details page, the exception indicator refreshes every five seconds (5000 ms). You can change this setting.

In the UTL_CONFIG_PARM table, you can specify a different (ms) PARM_VALUE for the SPEC2000_AEROBUY_ORDER_EXCEPTIONS_INDICATOR_POLLING configuration parameter.

Change adapter XML validation values

Incoming and outgoing XML messages are validated by default, but you can disable this validation.

In the UTL_CONFIG_PARM table, you can change the default values from `true` to `false` for the following rows:

PARM_NAME	PARM_VALUE *(default value)*
INCOMING_ADAPTER_XML_VALIDATION	`true`
OUTGOING_ADAPTER_XML_VALIDATION	`true`

Process long aircraft registration codes

If your aircraft registration code is longer than five characters, you can change the value of the SPEC2000_ACN_PROCESSING parameter from NONE to specify how the code should be processed for communicating with AeroBuy.

About this task

The ACN is a mandatory element of the ATA_SparesOrder message that communicates the three to five most important numbers of the aircraft registration code that are found in Maintenix on the Inventory Details page, Details tab, Aircraft Information area, Registration Code field.

You can specify one of the following values:

- LEFT_5_ALPHA_NUMERIC_CHARS - any dashes in the code are removed.
- RIGHT_5_ALPHA_NUMERIC_CHARS - any dashes in the code are removed.
- NUMERIC_CHARS - allows only allows numeric characters.
- NONE - the Registration Code is not changed. This is the default setting.

Steps

In the UTL_CONFIG_PARM table, for the SPEC2000_ACN_PROCESSING parameter, change the value.

Configure a custom integration user

If you change the default integration user in the `maintenix-config.properties` file, you must assign the INTEGRA role to the new user in Maintenix.

The default mxintegration system user is assigned the INTEGRA role. This role includes the API_RAISE_ALERT_REQUEST permission which is required to use the AeroBuy integration.

Note The default mxintegration user is different than the default mxspec2000pointegration user.

In the properties file you can see the user name and password, which you may choose to change depending on your Maintenix configuration:

```
mxserver.auth.username=mxintegration
  mxserver.auth.password=password
```

Enforce SPEC 2000 compliance

SPEC 2000 configuration parameters enable and disable validation of specific parameters to ensure SPEC 2000 compliance when you create an item in Maintenix. By setting these parameters in the UTL_CONFIG_PARM table, you can choose which rules to enforce.

SPEC2000_ACN_PROCESSING

Parameter Type: INTEGRATION

Configuration Type: GLOBAL

Category: Integration

Allowable Values: LEFT_5_ALPHA_NUMERIC_CHARS - Uses 5 alphanumeric chars from left and removes other chars like dashes, RIGHT_5_ALPHA_NUMERIC_CHARS - Uses 5 alphanumeric chars from right and removes other chars like dashes, NUMERIC_CHARS - Uses only numeric chars, NONE - Does not change the ACN

Default Value: NONE

Description: Determines a processing action for ACN in ATA_SparesOrder SPEC2000 message.

SPEC2000_AEROBUY_ORDER_EXCEPTIONS_INDICATOR_POLLING

Parameter Type: LOGIC

Configuration Type: GLOBAL

Category: Integration

Allowable Values: NUMBER

Default Value: 5000

Description: Determines the refresh frequency (ms) for the Order Exceptions Indicator.

SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Integration Message
Allowable Values: TRUE/FALSE
Default Value: TRUE
Description: Determines whether to add additional wrapper to outbound message.

SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER_ENVELOP

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Integration Message
Allowable Values: Aerobuy outbound message wrapper envelop.
Default Value: <aer:receiveRequest xmlns:aer="aeroxchange.com"><inXMLData><![CDATA[{0}]]></inXMLData></aer:receiveRequest>
Description: Aerobuy outbound message wrapper envelope.

SPEC2000_ALLOW_DUPLICATES

Parameter Type: SECURED_RESOURCE
Configuration Type: USER
Category: Admin - Spec 2000
Allowable Values: TRUE/FALSE
Default Value: FALSE
Description: Permission to allow duplicate values.

SPEC2000_CHECK_DUPLICATES

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: FALSE
Description: Enable duplicates check.

SPEC2000_ENFORCE_PART_NO_COMPLIANCE

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: FALSE
Description: If true, part numbers must be SPEC2000 compliant.

SPEC2000_MANUFACT_CD_ENABLE

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: FALSE
Description: Enable regex pattern for manufacturer.

SPEC2000_ORDER_NUM_LIMIT

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: TRUE/FALSE
Default Value: FALSE
Description: Order Barcode must not exceed nine characters.

SPEC2000_PO_PRIORITY_MAP

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: String pattern
Default Value: AOG-10,WSP-20,USR-40
Description: This parameter specifies the PO priority order (PRIORITY_ORD column in the REF_REQ_PRIORITY table) to the SPEC200 PRI element code. PRIORITY_ORD above the biggest mapped value will be considered RTN (Routine).The format is a comma-delimited list of code, dash, then maximum order for that code. The list must be specified in an incrementing order.

SPEC2000_UPPERCASE_ACCOUNT_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Account Code must be uppercase.

SPEC2000_UPPERCASE_ASSMBL_BOM_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic

Allowable Values: true/false
Default Value: TRUE
Description: Config slot code must be uppercase.

SPEC2000_UPPERCASE_ASSMBL_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Assembly Code must be uppercase.

SPEC2000_UPPERCASE_BOM_PART_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Part Group code must be uppercase.

SPEC2000_UPPERCASE_IETM_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: IETM Code must be uppercase.

SPEC2000_UPPERCASE_LABOUR_SKILL_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Labor Skill Code must be uppercase.

SPEC2000_UPPERCASE_LICENSE_TYPE_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false

Default Value: TRUE

Description: License Type Code must be uppercase.

SPEC2000_UPPERCASE_LOC_CD

Parameter Type: LOGIC

Configuration Type: GLOBAL

Category: Core Logic

Allowable Values: true/false

Default Value: TRUE

Description: Location Code must be uppercase.

SPEC2000_UPPERCASE_MANUFACT_CD

Parameter Type: LOGIC

Configuration Type: GLOBAL

Category: Core Logic

Allowable Values: true/false

Default Value: TRUE

Description: Manufacturer code must be uppercase.

SPEC2000_UPPERCASE_PANEL_CD

Parameter Type: LOGIC

Configuration Type: GLOBAL

Category: Core Logic

Allowable Values: true/false

Default Value: TRUE

Description: Panel Code must be uppercase.

SPEC2000_UPPERCASE_PART_NO_OEM

Parameter Type: LOGIC

Configuration Type: GLOBAL

Category: Core Logic

Allowable Values: true/false

Default Value: TRUE

Description: OEM Part Number must be uppercase.

SPEC2000_UPPERCASE_STOCK_NO_CD

Parameter Type: LOGIC

Configuration Type: GLOBAL

Category: Core Logic

Allowable Values: true/false

Default Value: TRUE

Description: Stock Code must be uppercase.

SPEC2000_UPPERCASE_TASK_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Task Code must be uppercase.

SPEC2000_UPPERCASE_VENDOR_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Vendor Code must be uppercase.

SPEC2000_UPPERCASE_ZONE_CD

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: TRUE
Description: Zone Code must be uppercase.

SPEC2000_VENDOR_CD_ENABLE

Parameter Type: LOGIC
Configuration Type: GLOBAL
Category: Core Logic
Allowable Values: true/false
Default Value: FALSE
Description: Enable regex pattern for vendor.

Update definition of the outbound message wrapper

You can update the envelope that AeroBuy messages are wrapped in when they are sent. This may be required, depending on your agreement with AeroXchange.

The SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER parameter is set to `true` by default, which means that outbound messages are wrapped with the wrapper that is specified in the SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER_ENVELOP parameter.

To change the envelope, update the SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER_ENVELOP parameter in the UTL_CONFIG_PARM table. The default value is:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"><soapenv:Body><aer:receiveRequest
  xmlns:aer="aeroxchange.com"><inXMLData><![CDATA[{0}]]></inXMLData></
aer:receiveRequest></soapenv:Body></soapenv:Envelope>
```

where `<![CDATA[{0}]]>` is replaced by the message content.

Troubleshooting your AeroBuy configuration

If you are experiencing issues with your AeroBuy configuration, the following solutions may resolve them.

Possible solutions to potential issues

Behavior	Possible Solution
PubSub purges transaction and event tables, but you receive no confirmation from AeroBuy.	Database is not configured with the correct SOAP envelope for AeroBuy OR the wrapper is set to `false`. Two rows in the database need corrections: • SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER must be `true`. • SPEC2000_AEROBUY_OUTBOUND_MESSAGE_WRAPPER_ENVELOPE should match the format you agreed upon with AeroXchange.
	AeroBuy account improperly configured. Contact AeroXchange to verify your account is properly set up.
AeroBuy returns a 101 error when a PO is sent.	Invalid Schema Error. There are missing fields in the PO or fields with the wrong data type.

Behavior	Possible Solution
AeroBuy returns a 403 error when a PO is sent.	AeroBuy does not recognize the CIC or SPL. Both fields must be registered in the AeroBuy database.
Messages are not reaching PubSub.	PubSub is not enabled. Check `pubsub.properties` to see if `enabled=true` is set for the configuration property.
	Malformed subscription. Check for the subscription in the PubSub console (`http://<server>/pubsub/console`). Check for errors in the service logs related to `pubsub-subscriptions.properties` or `mx-pubsub-subscriptions.properties`.
	Invalid subscription endpoint. Check `pubsub-subscriptions.properties` or `mx-pubsub-subscriptions.properties` for `subscription.<channel_name>...` entries.

	Invalid credentials.
	Check your subscription for invalid credentials in the `pubsub-subscriptions.properties` or `servers.properties` files. Look for typos and proper subscription formatting:
	`subscription.<channel_name>.<appid>.config .username=<username>`
	`subscription.<channel_name>.<appid>.config .password=${encrypted:<password>}`
	`subscription.<channel_name>.<appid>.url=<endpoint>`
	Then restart the server.

Behavior	Possible Solution
	Unbound variables. Verify that any variables such as `${some.var}` used in the subscriptions are defined earlier in the file or in the `servers.properties` file or the `mx-servers.properties` file. Check for errors in the server log that echo the variable names but not their values. For example: `HostNotFoundException - ${someserver} could not be reached`
	Destination server is experiencing an outage or is unreachable. Check the PubSub console (`http://<server>/pubsub/console`) for transactions that have processed as `false` and show multiple attempts.
	Destination server is returning errors. Check the PubSub console (`http://<server>/pubsub/console`) for transactions that have that have failed as `true`.

4

About PubSub

PubSub is an integration framework that allows for multiple systems to communicate without knowing about one another.

This communication is achieved by following a "publish - subscribe" model. In this model, one system, called the publisher, may publish a message to a specific channel while another system, the subscriber, may subscribe to that channel. The two systems do not know about one another; they only know about the PubSub framework, which sits in the middle, operating as a message broker.

For additional information on PubSub, such as the data model, the message queue, alerts, configuration, troubleshooting, and enabling or disabling, see the *Maintenix PubSub User Guide*.

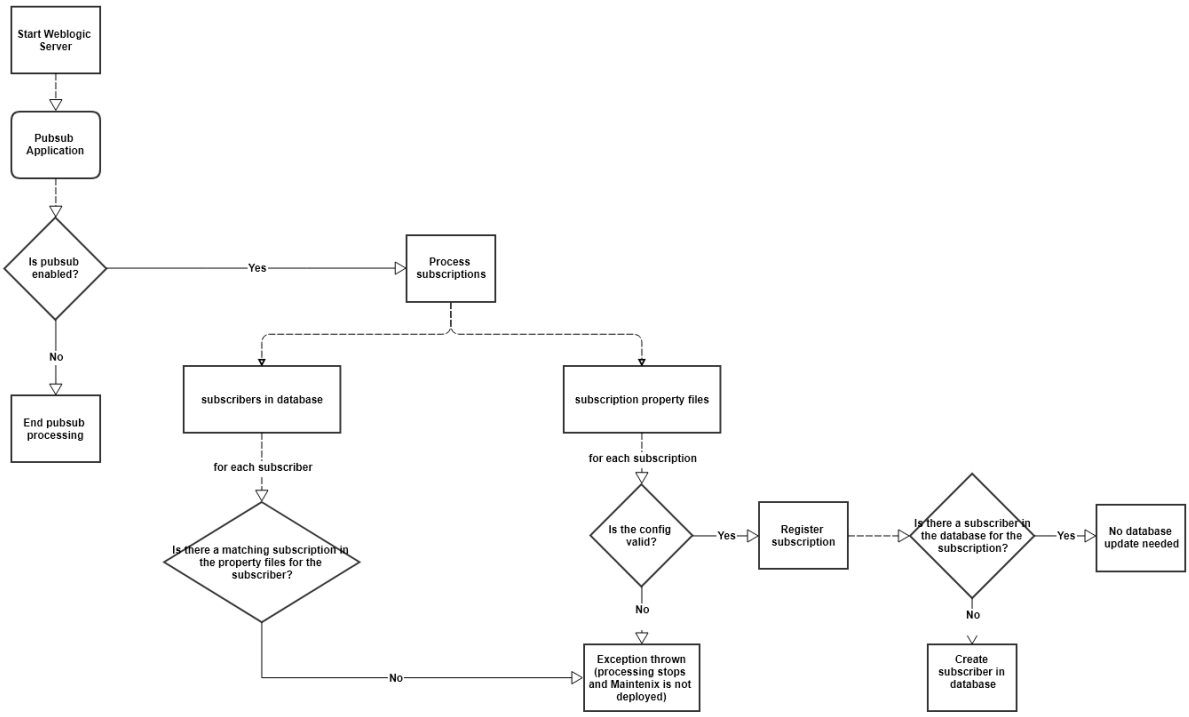
Subscribers

The subscriber is the PubSub component that receives messages on a specified subscription.

To be a subscriber in the PubSub framework, a system must provide the following information:

- The channel it wants to subscribe to.
- An endpoint for the PubSub framework to send messages to.
- Any necessary connection configuration information.

An application registering a subscriber must also have an ID for itself. A subscriber is uniquely identified by its channel and application ID.



Subscriber flow diagram

Registering subscribers

The PubSub framework exposes a few property files. Systems add entries to the property files to register themselves as subscribers and provide connection information. These entries are called subscriptions.

Example

On a server (server1), there is an application (app1) that wants to subscribe to a channel with the name TEST. The complete URL to the application is `http://server1/app1/endpoint`, where `http://server1` is the server URL.

To register the subscriber, the following property files must be updated:

- the `pubsub-subscriptions.properties` file:

```
subscription.TEST.app1.serverId=server1
subscription.TEST.app1.path=/app1/endpoint
```

- the `servers.properties` file:

```
server1.url=http://server1
```

When the server starts up, the subscriber is registered.

Registration exceptions

Certain exceptions during subscriber registration will cause the PubSub application to fail and hence the Maintenix deployment to fail. An error message is shown to indicate which subscriptions have problems.

These exceptions include:

- A subscriber is in the database but is not in the subscription property files.
- A subscription fails validation because of a missing mandatory property.
- A subscription fails validation because of a server property:
  - The server definition cannot be found (invalid server ID).
  - The server definition is missing a mandatory property.
  - The server definition has an invalid or unsupported URL protocol.

Registration warnings

Other scenarios during subscriber registration can cause a warning. The following warning does not prevent the subscriber from being registered or the PubSub application from starting:

- The subscriber has unrecognized properties

Subscriptions

Subscribers register for channels via subscriptions which follow specific format and are added to the `pubsub-subscriptions.properties` file. PubSub accepts customer subscriptions for HTTP and JMS endpoints.

Subscription format

All subscriptions must define a server ID which references a server defined in one of the server property files:

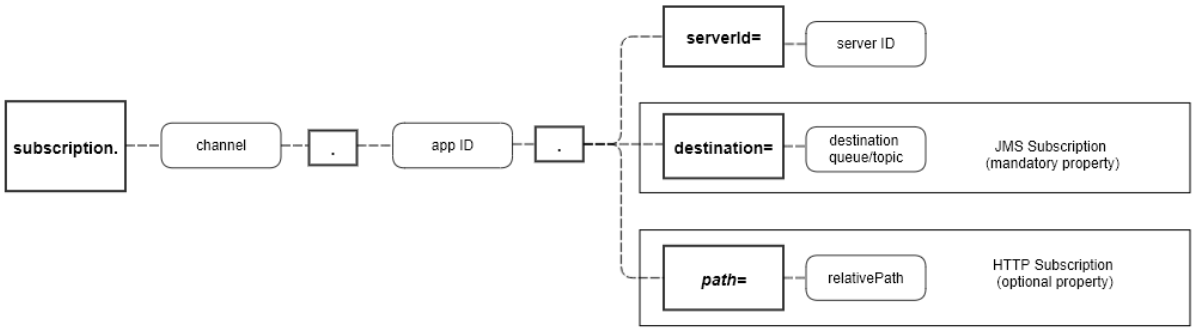
```
subscription.<channel>.<appId>.serverId=<serverId>
```

Subscriptions may also have an optional or mandatory configuration:

```
subscription.<channel>.<appId>.<property>=<value>
```

- The prefix `subscription` defines the start of a subscription property key.
- The field `<channel>` is replaced with the name of the channel to subscribe to.

- The field `<appId>` is replaced with the application identifier (together the channel and the app id make up the unique key for a subscription).
- The field `<property>` is replaced with a configuration property; (The configuration properties of the subscription are defined in the sections that follow.)



Subscription format

Defining server properties

All PubSub subscriptions must have a reference to a server through the server ID. This server can be defined in either `servers.properties` or `mx-servers.properties` files as follows.

- All servers must define a URL:

```
<serverId>.url=<url>
```

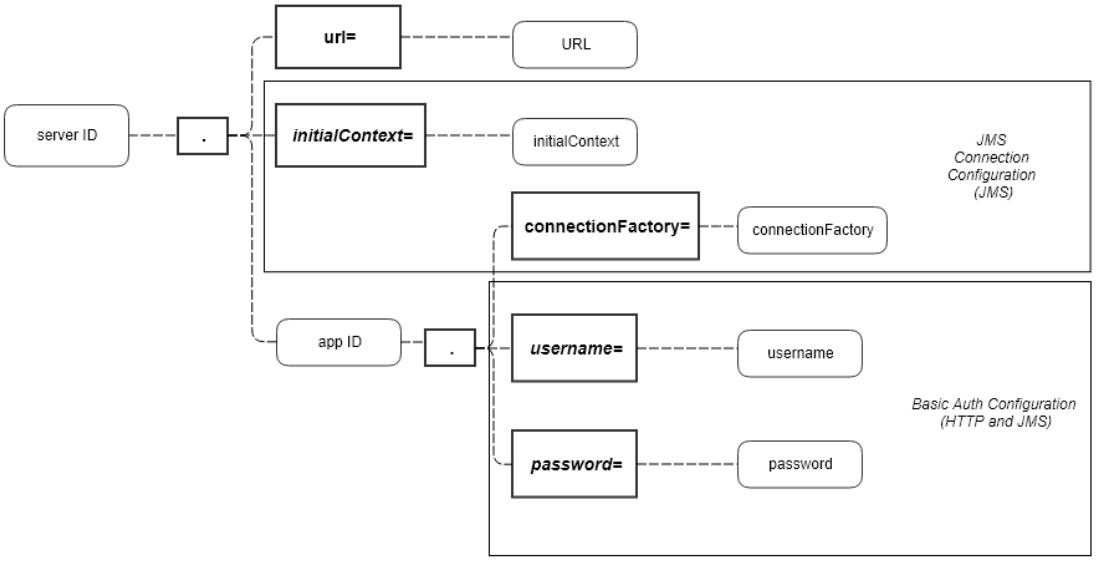
- The server may have its own properties:

```
<serverId>.<property>=<value>
```

- There may be many applications on a server with their own properties:

```
<serverId>.<appId>.<property>=<value>
```

- The prefix `<serverId>` is replaced with the name of the server ID.
- The field `<appId>` is replaced with the application identifier.



Server properties

HTTP subscriptions

Production use of htttp subscriptions consist of valid protocols, configurations, and structure.

Valid protocols for an HTTP subscription:

- http
- https

Allowable configuration:

- path (optional - specifies a relative path to append to the server ID's URL)

HTTP subscriptions refer to a server ID for the server defined in `mx-servers.properties` or `servers.properties` files.

Recommended structure for production use:

- In the `servers.properties` file, define the server information:
  - URL - The URL is the server URL (mandatory)
  - Credentials - The credentials are for a specific application of the server (optional)
- In the `pubsub-subscriptions.properties` file, point the subscription to the server:
  - The subscription gets its connection information from the server properties specified by the serverId (mandatory)
  - The subscription can also have a specific path to append to the server URL (optional)

- If credentials are required, ensure the application ID of the subscription matches the application ID used when specifying server credentials

Example subscription 1

in the `servers.properties` file:

```
server1.url=http://server1:8080
server1.app1.username=myUser
server1.app1.password=${encrypted:myPassword}
```

in the `pubsub-subscriptions.properties` file:

```
subscription.TEST.app1.serverId=server1
subscription.TEST.app1.path=/relative/path
```

Example subscription 2

There are some pre-defined servers and authentication URLs in the `mx-servers.properties` file. More information about these is provided in the integration guides that use these properties. PubSub has internal knowledge of which pre-defined servers use which authentication scheme.

For example, the eLogbook integration uses Comply365 for authentication, and so we see eLogbook defined as a server property, as well as a property for the Comply365 authentication server.

In the `mx-servers.properties` file:

```
comply365Auth.url=https://ifs-test-api.comply365.net
eLogbook.url=https://ifs-test-api.comply365.net
eLogbook.eLogbookApp.username=mxelogbook
eLogbook.eLogbookApp.password=${encrypted:no1knows}
```

The values (for example, the eLogbook server URL) must be filled in by the customer after installation.

In the `pubsub-subscriptions.properties` file:

```
subscription.ELOGBOOK_CHANNEL.eLogbookApp.serverId=eLogbook
```

JMS subscriptions

This topic covers valid protocols, configuration, and structure for production use of JMS subscriptions. It also lists a few examples of JMS subscriptions.

Valid protocols for a JMS subscription:

- t3
- ssl
- tcp

Allowable configuration:

- destination (mandatory - specifies the queue or topic to append to the JMS server URL)

JMS subscriptions refer to a server ID for the server defined in the `mx-servers.properties` or the `servers.properties` files. Maintenix Weblogic has default server properties already defined in the `mx-servers.properties` file to facilitate adding subscriptions to Weblogic queues and topics.

Recommended structure for production use:

- In the `servers.properties` file, define the following information:
  - URL - the URL is the server URL (mandatory).
  - Connection factory - the connection factory is the Java class to use as a connection factory and is specific to an application on the server (mandatory).
  - Initial context - the initial context holds a list of properties defining the initial context for JNDI which is used to look up the connection factory (optional). Note that the list is comma-separated.
  - Credentials - The credentials are for a specific application of the server (optional).
- In the `pubsub-subscriptions.properties` file, point the subscription to the server:
  - The subscription will get its connection information from the server properties specified by the serverId (mandatory).
  - The subscription also must have a destination queue or topic to append to the server URL (mandatory).
  - If credentials are required, ensure the application ID of the subscription matches the application ID used when specifying server credentials.

Multiple URLs format:

If you have multiple URLs, separate them by commas:

```
server1.url=<url1, url2>
```

where `<url1, url2>` is a list of URLs.

Example 1 - Minimal queue subscription with Maintenix Weblogic

The Maintenix Weblogic server properties are defined in the `mx-servers.properties` file with a default application:

```
maintenix.weblogic.url=t3://localhost:7001
```

```
maintenix.weblogic.initialContext=java.naming.factory.initial=weblogic.
 jndi.WL
InitialContextFactory
maintenix.weblogic.default.connectionFactory=com.mxi.am.
 jms.ConnectionFactory
```

Add a subscription to the `mx-pubsub-subscriptions.properties` file:

```
subscription.TEST.default.serverId=maintenix.weblogic
subscription.TEST.default.destination=myQueue
```

Example 2 - Queue subscription with basic Auth and Weblogic context

In the `servers.properties` file:

```
server1.url=t3://localhost:7001/
server1.initialContext=java.naming.factory.initial=weblogic.jndi.
 WLInitialCont
extFactory
server1.app1.connectionFactory=com.mxi.am.jms.ConnectionFactory
server1.app1.username=myUser
server1.app1.password=${encrypted:myPassword}
```

In the `pubsub-subscriptions.properties` file:

```
subscription.TEST.app1.serverId=server1
subscription.TEST.app1.destination=myQueue
```

Example 3 - Topic subscription with multiple URLs and Tibco SSL context

In the `servers.properties` file:

```
server1.url=ssl://localhost:7001,ssl://localhost:7002/
server1.initialContext=java.naming.factory.initial=com.tibco.
 tibjms.naming.Tib
jmsInitialContextFactory,com.tibco.tibjms.naming.
 security_protocol=ssl,com.tib
co.tibjms.naming.ssl_enable_verify_host=false
server1.app1.connectionFactory=com.mxi.am.jms.ConnectionFactory
server1.app1.username=myUser
server1.app1.password=${encrypted:myPassword}
```

In the `pubsub-subscriptions.properties` file:

```
subscription.TEST.app1.serverId=server1
subscription.TEST.app1.destination=myTopic
```

PubSub property files and subscriptions

There are four property files that PubSub scans for subscription information: `pubsub-subscriptions.properties, servers.properties, mx-pubsub-subscriptions.properties, mx-servers.properties`. They contain either Maintenix-specific information or customer-specific information and variables for subscriptions.

The PubSub property files fall in two categories: customer files (`pubsub-subscriptions.properties` and `servers.properties`) and Maintenix files (`mx-pubsub-subscriptions.properties` and `mx-servers.properties`).

Customer files

The content of the customer-owned files is preserved between upgrades.

pubsub-subscriptions.properties

Contains customer-owned subscriptions and variables for subscriptions.

This file is the primary mechanism for adding subscribers.

servers.properties

Contains variables for customer server configuration

Maintenix files

The content of the Maintenix files is owned by IFS and is subject to change between upgrades.

mx-pubsub-subscriptions.properties

Contains Maintenix-specific subscriptions

mx-servers.properties

Contains variables for Maintenix server configuration

Editing the properties files

- Any additions by customers should be done in the customer-owned files, over which the customer has full control.
- Customers may modify the values of the `mx-servers.properties` file, but they should not add or remove keys.
- No modifications should be made to the `mx-pubsub-subscriptions.properties` file.

Important It is recommended to keep the server-configuration variables in the `servers.properties` file to facilitate the move from test to production environment.

The Maintenix `mx-servers.properties` and `servers.properties` files store dynamic information that will change between environments (e.g., host and port information).

The content of the `mx-pubsub-subscriptions.properties` and `pubsub-subscription.properties` files should store static information. If these standards are followed, then the only files that will need to be updated in an environment upgrade will be the `servers.properties` and `mx-servers.properties` files.

Important Having duplicated variables across the property files or within one property file is not recommended; it will cause errors on startup or when dispatching a message to a subscriber.

Variables

You can create variables that can later be used in defining subscriptions. Variables can be defined in either of the customer-owned property files: `pubsub-subscriptions.properties` or `servers.properties`.

Variable interpolation

Variables are very useful for containing content that may change between environments. Variables can also be reused between many subscriptions (for example, a queue name).

For example:

- In the `servers.properties` file, you can define the following variable:

```
server1.myqueue=myQueue
```

- Then in `pubsub-subscriptions.properties` file, you can refer to that variable as:

```
subscription.TEST.app1.destination=${server1.myqueue}
```

Best practices for variable naming

Variable names must be unique, or you will get exceptions when starting up of the server. You are recommended to use a distinctive name for each variable.

Lookups

PubSub supports two types of lookups that allow specialized processing of a value: a custom "encryption" lookup to encrypt properties, and the CyberArk lookup.

The lookup format is similar to that of variable interpolation but has a keyword followed by a colon (:).

For example:

```
app1.key=${lookup_keyword:lookup_value}
```

Important Nested variable interpolations inside lookups are not supported. In the preceding example, replacing `lookup_value` with a variable like `${lookup_variable}` would be invalid.

Property Encryption

There are two methods of encrypting values in the property file: using an encrypted lookup or using the .password suffix on a property key.

- Example of using an encrypted lookup:
  - How to use it: `app1.key=${encrypted:valueToEncryptGoesHere}`
  - After encryption:

`app1.key=${encrypted:[AES]ulWDBAJ7lARKEnLCkzDGIA==}`

- Example of using the `.password` suffix on a property key.
- The property key must end with ".password" and the value must be plain-text.
  - How to use it: `app1.key.password=valueToEncryptGoesHere`
  - After encryption:

`app1.key.password=${encrypted:[AES]ulWDBAJ7lARKEnLCkzDGIA==}`

When the server starts up for the first time, the property files will be scanned and any values that need encryption (as indicated by one of the two methods above) will be encrypted. The value in the file will be rewritten to the encrypted value that only the server can decrypt.

Important You cannot use variable interpolation with either encryption method.

Note When setting a password in the text editor, the password will be encrypted when the server is restarted. To have the password automatically encrypted as you enter it, use the Password Encryption Utility. See the *Maintenix Installation and Configuration Guide* for more details.

Examples of invalid formats

Example 1:

```
someVariable=valueToEncryptGoesHere
```

In this example, nothing will be encrypted

Example 2

```
app1.key=${encrypted:${someVariable}}
```

In this example, `"${someVariable}"` will be treated as the value and encrypted

Example 3

```
someVariable=valueToEncryptGoesHere
app1.key.password=${someVariable}
```

This example is not technically invalid, but it illustrates the risk of a password remaining unencrypted. Here `${someVariable}` will be replaced by its value `"valueToEncryptGoesHere"` which is never encrypted.

CyberArk

By implementing a CyberArk lookup, PubSub supports CyberArk for those customers that have CyberArk support installed. CyberArk is a password vault system.

The following string is an example of the CyberArk lookup format:

```
${cyberark:<safe>:<folder>:<object>:<reason>:<app_id>}
```

An example below displays a subscription using CyberArk in the `servers.properties` file:

```
server1.url=http://localhost:7001
server1.app1.username=myUser
```

```
server1.app1.password=${cyberark:safe1:folder1:object1:reason1:app1}
```

An example in the `pubsub-subscriptions.properties` file:

```
subscription.TEST.app1.serverId=server1
subscription.TEST.app1.path=/endpoint
```

This example illustrates the use of CyberArk lookup. Note that there is a ':' separated ordered list of information that must be communicated with CyberArk to allow it to provide PubSub with the password when dispatching a message.

The list order is as follows:

```
safe, folder name, object, reason code, app id
```

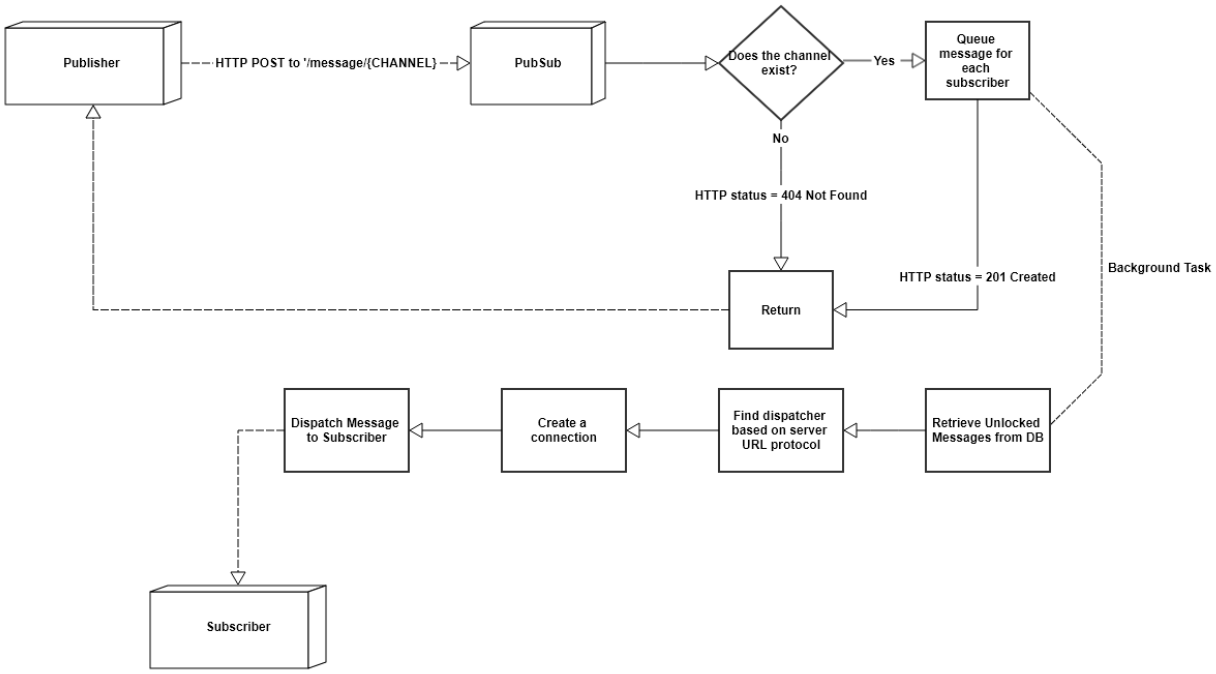
Publishers

Unlike subscribers, publishers are not registered using the property file. Publishing systems simply send a message to the PubSub framework and nothing more. The PubSub framework makes no attempts to transform the message sent by the publisher; it simply dispatches the message to all subscribers of the channel on which the message was sent.

The following endpoint exists to publish a message to PubSub:

- HTTP endpoint `<server>/pubsub/message/<channel>`

Note If the message was sent to a channel that doesn't exist, then the message is dropped and a "404 Not Found" response is returned.



Publisher flow diagram

Properties/Headers

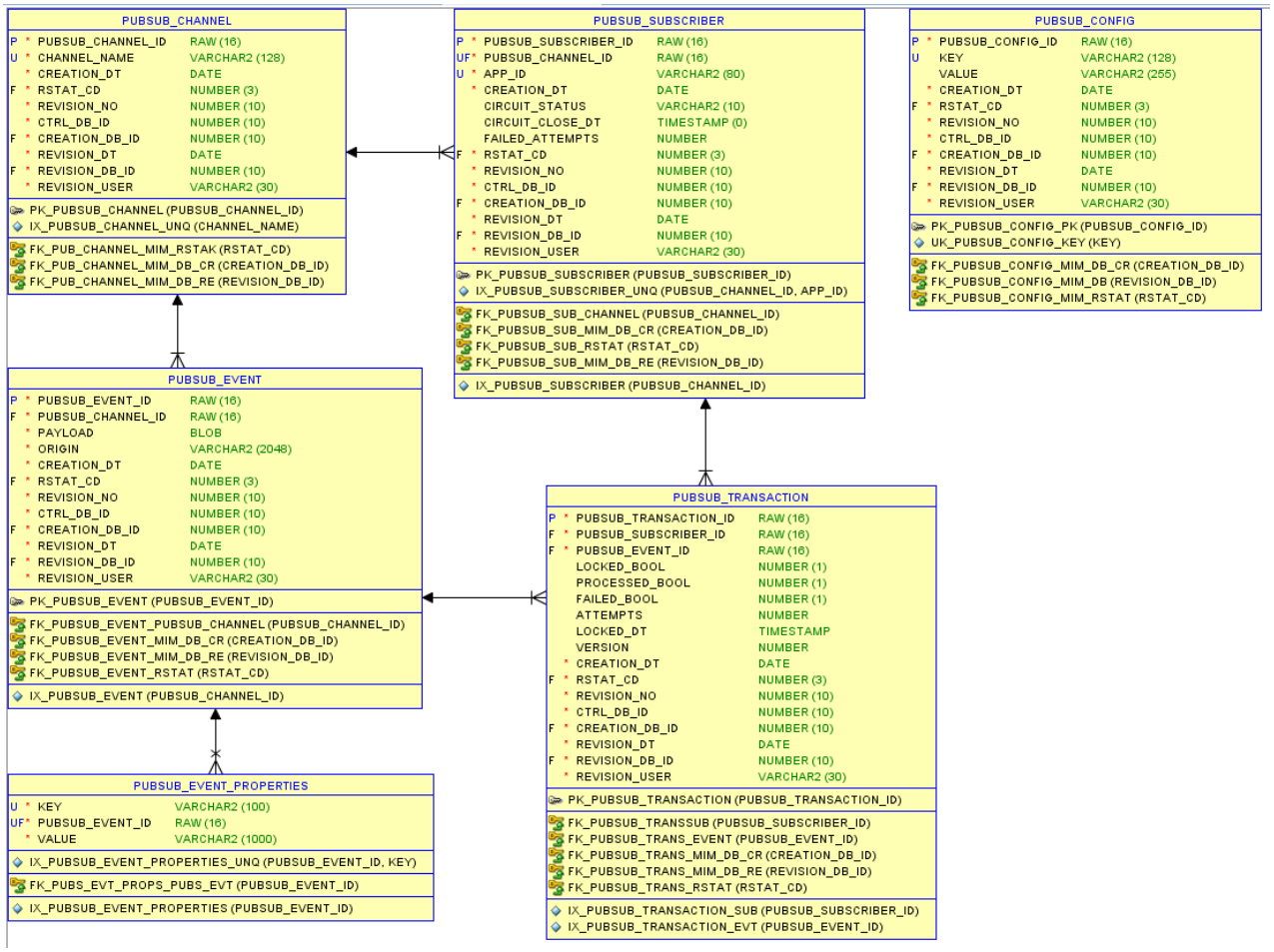
When the message is dispatched (via HTTP or JMS), any property/header keys that are prefixed with an "x-" or "X-" are passed along to the subscriber. The HTTP dispatcher will also pass along the Content-Type header.

Message body

Whatever the publisher sends as the message body is what the subscriber will receive. This content may be plain text, JSON, XML, YAML, etc. The PubSub framework DOES NOT perform any message transformation.

Data model

The following figure shows the database model for the PubSub framework.



The purpose of the PUBSUB_EVENT table is to store information about a message, such as its content and which channel it belongs to; however, it does not directly join to the subscribers.

This connection occurs through the PUBSUB_TRANSACTION table, which stores information about the publishing of a message to a specific subscriber. This table tracks whether the message has been successfully processed, and how many attempts have been made to deliver the message.

The message queue

The message queue is what drives the PubSub framework under the hood. It is responsible for loading messages from the database and processing them.

This process occurs every five seconds and is executed on multiple threads to improve execution time.

Clustered environments

Due to the threaded nature of the message queue and the potential for the PubSub framework to be deployed in a clustered environment, it is important to understand how messages are processed.

The PUBSUB_TRANSACTION table has a column named LOCKED_BOOL. This Boolean value is used to track whether or not the PubSub transaction is currently being processed. When the message queue runs, it gathers all transactions that have a value of false for LOCKED_BOOL and PROCESSED_BOOL. These unlocked transactions are then locked to prevent any other PubSub servers from picking them up, which in turn prevents messages from being delivered multiple times.

If a message is successfully delivered, then its PROCESSED_BOOL value is set to TRUE and is eligible to be deleted. It is important to note that when a transaction is locked, its LOCKED_DT value is updated. If a transaction has not been successfully processed and has a LOCKED_DT that is more than 30 minutes in the past, it will count as having a LOCKED_BOOL value of FALSE. This process is in place to handle the scenario in which one of the servers in the cluster has gone down while processing a message.

Known Limitation: In a clustered environment, the configuration will be applied only to the node handling the REST request. All other nodes will not have their configuration updated until their application servers have been restarted.

Note If a transaction is successfully processed, it will be automatically deleted by the message queue from the database. If all transactions for a message are deleted, then that event will also be deleted from the database.

Alerts

PubSub raises alerts to Maintenix to indicate to users when there is a communication error or another scenario that needs to indicate to users when there is a communication error or another scenario that needs attention.

There are two instances when PubSub raises alerts:

- A message fails to publish and is not retried.
- A subscribers circuit is closed due to too many retry attempts.

By default, the alerts are assigned to the INTEGRA role. PubSub alerts are managed in Maintenix on the Alert Setup page. For more information on alerts, see the Maintenix Administration Guide.

PubSub alert values

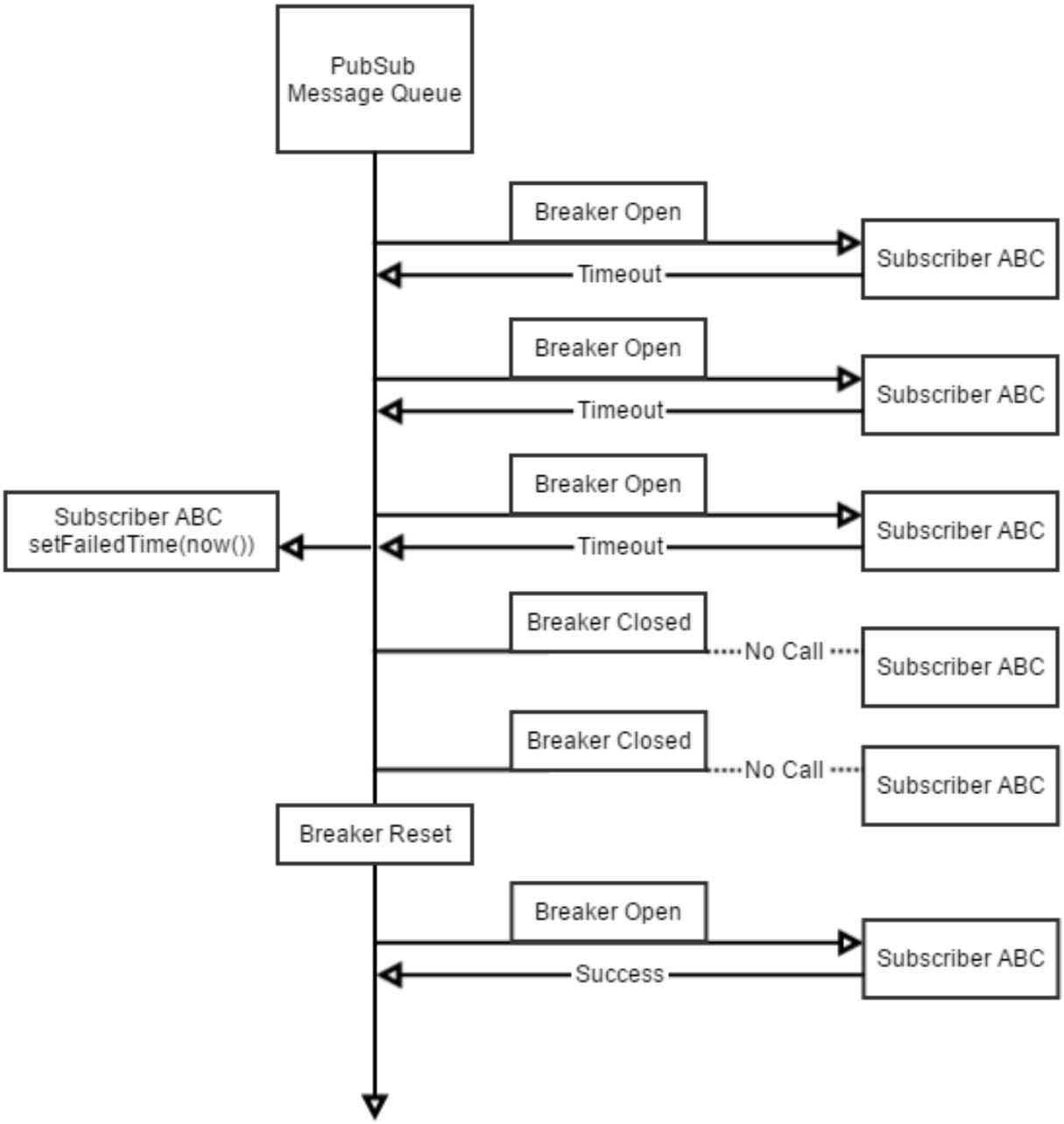
Alert Name	Role	Description	Message
PubSub subscriber circuit closed	INTEGRA	No messages will be sent to the subscriber while its circuit is closed.	`The circuit was shutdown for the subscriber with app ID [appId] and channel [channel] due to connection issues on the server with ID [serverId]. No messages will be sent to this subscriber until the circuit reopens. See the server logs for more information.`
PubSub transaction failed	INTEGRA	Failed to publish message to PubSub channel.	`Failed to publish message to channel [channel] with subscriber with app ID [appId] for the transaction with ID [transactionID]. Error: [error_message]`

Guaranteed delivery and handling failures

The PubSub system guarantees message delivery by implementing the circuit breaker pattern.

When the system receives a message, it will create a transaction for each subscriber and attempt to dispatch that transaction to the subscriber.

The following diagram outlines the basic idea of the circuit breaker pattern in the context of the event queue.



Circuit breaker diagram

Handling failures

PubSub has measures for handling message failure.

If the system fails to dispatch the message, it will do one of two things:

1. If the failure is deemed to be recoverable, the fail counter will be incremented on the subscriber, and the transaction will be tried again.

2. If the failure is deemed to be non-recoverable, the transaction will be marked as failed which prevents any further attempts on the transaction.

If a subscriber's fail count passes the configured threshold, then the subscriber's circuit will be shut down for the configured amount of time. While the circuit is shut down, no attempts will be made to dispatch messages to the subscriber. This process helps to prevent the PubSub system from becoming overloaded by failing messages. After the configured shutdown duration has passed, the circuit will reopen and more delivery attempts to the subscriber will be made.

Note While PubSub guarantees message delivery, it is possible in certain circumstances that a message is delivered multiple times to the same subscriber. Please be aware of this and take appropriate precautions even though it is an infrequent occurrence.

JMS failures

If a JMS `ConnectException` is thrown, it is considered recoverable and the transaction will be tried again. Any other exception that occurs during dispatching a message via JMS is considered non-recoverable.

Non-recoverable exceptions include:

- The given connection factory cannot be found.
- The specified queue or topic cannot be found.

HTTP failures

If an HTTP connection cannot be established due to a connection exception, this event is treated as a recoverable exception. If any other type of exception occurs before dispatching, it is considered non-recoverable.

If a connection is successfully established, then the classification of recoverable or non-recoverable depends on the HTTP status code returned in the HTTP response.

Recoverable HTTP Status Codes

The following HTTP response status codes are considered recoverable by PubSub:

- 408 (Request Timeout)
- 429 (Too Many Requests)
- 503 (Service Unavailable)
- 504 (Gateway Timeout)
- 507 (Insufficient Storage)
- 509 (Bandwidth Limit Exceeded)
- 598 (Network read timeout error)

- 599 (Network connect timeout error)

Non-recoverable HTTP Status Codes

HTTP response status codes above 299 (300 and higher) are considered non-recoverable, so they do not appear on the recoverable list. For example, 500 (Internal Server Error).

How to retry failed transactions

To retry a failed transaction, you can manually update the pubsub_transaction table in the Maintenix database. Set both the failed_bool and locked_bool to 0 to reset the transaction.

For example, use the following SQL to reset all transactions:

```
UPDATE
 pubsub_transaction
 SET
 failed_bool = 0,
 locked_bool = 0
 WHERE
 failed_bool = 1;
```

Dispatchers

Before a message can be delivered to a subscriber, it must first be dispatched. This is achieved through the use of dispatchers. Each dispatcher is responsible for delivering a message to a subscriber based on the subscriber's URL.

Each dispatcher may support one or more protocols. These protocols are used to determine which dispatcher should dispatch the message to the subscriber.

For example, the HTTP dispatcher supports the HTTP and HTTPS protocols. Given a subscription with the URL endpoint set as `http://system1/endpoint`, the protocol will be identified as HTTP and the PubSub framework will dispatch the message using the HTTP dispatcher. If you choose to use the JMS dispatcher, you will need some additional configuration. Here is an example of a JMS subscription:

- in the `servers.properties` file:

```
jmsServer1.url=t3://localhost:7001/
jmsServer1.jmsServer1App.username=system
jmsServer1.jmsServer1App.password=${encrypted:password1}
jmsServer1.jmsServer1App.connectionFactory=com.mxi.am.jms.
 ConnectionFactory
jmsServer1.initialContext=java.naming.factory.initial=weblogic.
 jndi.WLInitial
ContextFactory
```

- in the `pubsub-subscriptions.properties` file:

```
subscription.TEST2.jmsServer1App.serverId=jmsServer1
subscription.TEST2.jmsServer1App.destination=someQueue
```

The dispatcher requires a connection factory class with the fully qualified Java class name. Also, there are JNDI initial context parameters specified as a comma-separated list of keys and values (key=value). The JNDI initial context is used to look up the connection factory.

PubSub configuration

The PubSub system can be configured through a REST endpoint located at `sys/config` and also through the WebLogic system management console.

The REST resource exposes a number of variables which affect the behavior of the PubSub system. Some variables determine their value based on the current state of the system, while others are simply values (all of which have a default). When a client sets a config value, that value is stored in the database. The next time the PubSub system starts up, it will load the values from the database instead of using the defaults.

The following table lists the configuration values that are currently exposed.

Key	Default	Description
QUEUE_STATUS	STARTED	This value displays and controls the status of the PubSub system's internal message queue. It may be set to a value of STARTED or STOPPED, however it may also return STARTING and STOPPING as values to better reflect the state of the system. Note that other configuration values may restart the queue, so that their new values can take effect.
QUEUE_CYCLE_TIME	5000	This value (time provided in milliseconds) represents how often the message queue runs. Changing this value will trigger a message queue restart.

Key	Default	Description
DELETE_PROCESS_STATUS	STARTED	This value displays and controls the status of the PubSub system's internal delete queue which cleans up processed transactions or events. It may be set to a value of STARTED or STOPPED, however it may also return STARTING and STOPPING as values to better reflect the state of the system. Note that other configuration values may restart the queue so that their new values can take effect.
DELETE_CYCLE_TIME	60000	This value (time provided in milliseconds) represents how often the delete queue runs. Changing this value will trigger a delete queue restart.
LOCK_EXPIRE_TIME	1800000	This value (time provided in milliseconds) represents the threshold for counting a locked transaction as unlocked.
SHUTDOWN_MODE	GRACEFUL	This value controls how thread pools are shut down. It may have a value of GRACEFUL or FORCE. GRACEFUL will wait for all threads in the pool to finish execution, regardless of how long it takes while FORCE will shut down the thread pool immediately.

Key	Default	Description
CIRCUIT_RESET_TIME	30000	This value (time provided in milliseconds) controls for how long a subscriber's circuit is in shutdown. When a subscriber's circuit is shutdown, no messages will be delivered to the subscriber. Once the specified time has elapsed, then the circuit will be reopened and messages will begin to flow.
DISPATCH_ATTEMPTS	5	This value controls how many connection errors must occur before a subscriber's circuit is shutdown.
BATCH_SIZE	100	This value controls how many unlocked transactions (or transactions with expired lock) are fetched to be dispatched to their subscribers at one time.
SEQUENCE_MAX	199	This value is the maximum value of the PUBSUB_BATCH_SEQ cyclical database sequence. The sequence ensures that multiple PubSub nodes do not dispatch the same messages. If the sequence maximum is updated in the database then this configuration parameter must also be updated to match it, and vice versa, in order for the sequence to be reset when no messages are found at later sequence values.

Certain values can be adjusted using the WebLogic system management console (`http://servername:7001/console`). The dispatcher queue has a thread pool with minimum and

maximum values depending on the load. These values can be adjusted in the console in the following location:

`Domain Structure > Deployments > assetmanagement (expand with the +) > pubsub > Configuration > Workload > wm/PubSub-DispatcherWorkManager`

When you navigate to this page, you will see two links at the bottom of the page for `PubSub-DispatchMaxThreadConstraint and PubSub-DispatchMinThreadConstraint`. You can navigate to each place and adjust the values. A server restart is required for the changes to take effect. The default is for a minimum of 5 threads and a maximum of 25 threads.

Enabling and disabling PubSub

PubSub is disabled by default. In a disabled state, PubSub will not publish messages to the subscribers.

You can still access and use the PubSub APIs, but a transaction (sending a message to a subscriber) will never be completed as the background processes for dispatching and deleting messages are turned off. Any configuration information in the database will be disregarded. To enable PubSub, find the `pubsub.properties` property file and set the value of `com.mxi.pubsub.config.enabled` to TRUE. Save the file, and restart the application server.

Troubleshooting PubSub

In most cases, the logs contain a message when an error occurs with PubSub. If there are no errors, it is likely because PubSub is turned off or the endpoint is improperly configured.

Scenario: Messages are not published to PubSub

Symptoms: There are no new entries in the pubsub_transaction and pubsub_event table right after an action occurs which should publish a message.

Possible Causes:

- PubSub is not turned on (`pubsub.properties -> com.mxi.pubsub.config.enabled=true`).
- The message is published by a trigger and that trigger is not enabled in utl_trigger.

Scenario: Messages are published to PubSub and are being sent out, but the endpoint is not receiving them

Symptoms: There are new entries in the pubsub_transaction and pubsub_event table right after an action occurs which publishes a message. The transactions are then processed and purged, but the endpoint does not receive a message.

Possible Causes:

- The endpoint is sending back a 200 for rejection/error scenarios. PubSub assumes the message has been accepted if it receives a response in the 200s. (If the message has a wrapper, ensure that the wrapper designated for the endpoint is used.)
- The endpoint account is improperly configured.

Scenario: Messages are published to PubSub, and PubSub is receiving error responses when it sends the message out

Symptoms: PubSub is not purging messages or transactions from tables. Transactions are marked as failed or have been attempted multiple times. There should be errors in the logs.

Possible Causes:

- Malformed subscriptions.
  - Wrong endpoint (check the subscription and server property files to see which server is being used and what the server's URL is).
  - Invalid credentials or authentication (username or password; check the server property files for credentials).
- Invalid schema is followed.
- Invalid message wrapper is used for the endpoint.
- Endpoint provided is not configured or active.

A

Sample messages

The XML messages that are used to communicate between Maintenix, AeroBuy, and vendor systems are defined by the Spec 2000 E-Business Specification for Materiels Management Guide, Revision 2014.1 (20th Revision).

Sample messages in .xml file format can be found in the `maitnenix\documentation \Integrators and Developers\API Sample Messages` folder.

SOAP envelope

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/
  envelope/"xmlns:aer="aeroxchange.com">
   <soapenv:Header/>
   <soapenv:Body>
      <aer:receiveKALRequest>
         <inXMLData>
            <![CDATA[ --XML message goes here-- ]]>
         </inXMLData>
      </aer:receiveKALRequest>
   </soapenv:Body>
</soapenv:Envelope>
```

ATA_SparesQuoteRequest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesQuoteRequest xmlns:xsi="http://www.w3.org/2001/
XMLSchema-

 instance"xsi:noNamespaceSchemaLocation="ATA_SparesQuoteRequest.xsd"
 version="1.00">
      <RequestHeader>
```

```
         <CIC>ACN</CIC>
         <SPL>98124</SPL>
      </RequestHeader>
      <QTR>ORB25834</QTR>
      <RequestDetails>
         <PNR>DAC29BBS0743</PNR>
         <QTQ UNT="EA">10</QTQ>
      </RequestDetails>
   </ATA_SparesQuoteRequest>
```

ATA_SparesQuoteInterim.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesQuoteInterim xmlns:xsi="http://www.w3.org/2001/
XMLSchema-

 instance"xsi:noNamespaceSchemaLocation="ATA_SparesQuoteInterim.xsd"
 version="1.00">
      <InterimHeader>
         <CIC>ACN</CIC>
         <SPL>98124</SPL>
      </InterimHeader>
      <QTR>ORB25834</QTR>
      <InterimDetails>
         <PNR>DAC29BBS0743</PNR>
         <REM>Will Advise 2005-03-21</REM>
      </InterimDetails>
   </ATA_SparesQuoteInterim>
```

ATA_SparesQuoteFinal.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesQuoteFinal xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"xsi:noNamespaceSchemaLocation="ATA_SparesQuoteFinal.xsd"
 version="1.00">
      <FinalHeader>
         <CIC>ACN</CIC>
         <SPL>98124</SPL>
      </FinalHeader>
      <QTR>ORB25834</QTR>
      <FinalDetails>
         <IPN>DAC29BBS0743</IPN>
         <RPN>DAC29BBS0745</RPN>
         <QTQ UNT="EA">123</QTQ>
         <KWD>BEARING</KWD>
         <UNP ICR="USD">103.00</UNP>
         <PFT>90</PFT>
         <LTM>01</LTM>
```

```
         <QTN>041957364</QTN>
         <QTD>2004-09-15</QTD>
      </FinalDetails>
   </ATA_SparesQuoteFinal>
```

ATA_SparesOrder.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesOrder version="1.01">
      <OrderHeader>
         <CIC>MXI</CIC>
         <SPL>CAGE1</SPL>
         <MessageSender>Mxi Technologies</MessageSender>
      </OrderHeader>
      <OrderDetails>
         <TNC>
            <TNC_Order>3</TNC_Order>
            <TNC_Item>4</TNC_Item>
            <TNC_Action>3</TNC_Action>
         </TNC>
         <CPO>P0100031-1</CPO>
         <PNR>A0000009</PNR>
         <UNP ICR="USD" UNT="EA">500.0</UNP>
         <QTO UNT="EA">1</QTO>
         <SSD>2016-04-19-04:00</SSD>
         <MFR>00001</MFR>
         <SSM>
            <ShippingMethodCode>AF</ShippingMethodCode>
            <ShippingPaymentMethodCode>A</ShippingPaymentMethodCode>
         </SSM>
         <SHT>ATL</SHT>
         <PRI>AOG</PRI>
         <ACN>UNK</ACN>
      </OrderDetails>
   </ATA_SparesOrder>
```

ATA_SparesMessageAck.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesMessageAck version="1.0">
      <MessageAckHeader>
         <CIC>CMP</CIC>
         <SPL>CAGE1</SPL>
         <MessageReceiver>superuser</MessageReceiver>
      </MessageAckHeader>
      <MessageTypeIndicator>Order</MessageTypeIndicator>
      <CPO>P0104355-1</CPO>
   </ATA_SparesMessageAck>
```

ATA_SparesOrderExc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesOrderExc version="1.12">
      <OrderExcHeader>
         <CIC>CMP</CIC>
         <SPL>CAGE1</SPL>
         <MessageSender>Allen Welsh</MessageSender>
         <MessageReceiver>superuser</MessageReceiver>
      </OrderExcHeader>
      <OrderExcDetails>
         <CPO>P0103919-1</CPO>
         <PNR>314-2200-18</PNR>
         <UNP ICR="USD" UNT="EA">140</UNP>
         <MFR>73030</MFR>
         <REM>1st Part Line. Vendor change to Price.</REM>
      </OrderExcDetails>
      <OrderExcDetails>
         <CPO>P0103919-2</CPO>
         <PNR>7517900-55004</PNR>
         <UNP ICR="USD" UNT="EA">160</UNP>
         <MFR>73030</MFR>
         <REM>2nd Part Line. Vendor change to Price.</REM>
      </OrderExcDetails>
   </ATA_SparesOrderExc>
```

ATA_SparesSupplierShipNotice.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesSupplierShipNotice version="1.12">
         <ShipNoticeHeader>
            <CIC>CMP</CIC>
            <SPL>CAGE1</SPL>
            <!--Optional:-->
            <MessageSender>Allen Welsh</MessageSender>
            <!--Optional:-->
            <MessageReceiver>superuser</MessageReceiver>
         </ShipNoticeHeader>
         <!--1 or more repetitions:-->
         <ShipNoticeDetails>
            <CPO>P0103941-1</CPO>
            <PNR>1001070-2</PNR>
            <SHQ UNT="EA">6</SHQ>
            <SHT>12345</SHT>
            <SHD>2087-08-08</SHD>
            <!--You have a CHOICE of the next 2 items at this level-->
            <ShipNoticeShipment>
               <BOL>12148794</BOL>
               <Carton>
                  <CartonDimensions UNT="CM">
                     <LEN UNT="CM">12</LEN>
                     <WID UNT="CM">12</WID>
                     <HGT UNT="CM">12</HGT>
                  </CartonDimensions>
```

```
                  <CartonWeight UNT="KG">100</CartonWeight>
               </Carton>
            </ShipNoticeShipment>
            <!--Optional:-->
            <PRI>RTN</PRI>
            <!--Optional:-->
            <CAR>FedEx</CAR>
         </ShipNoticeDetails>
      </ATA_SparesSupplierShipNotice>
```

ATA_SparesInvoice.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_SparesInvoice version="1.0">
      <InvoiceHeader>
         <CIC>CMP</CIC>
         <SPL>CAGE1</SPL>
         <MessageSender>Allen Welsh</MessageSender>
      </InvoiceHeader>
      <InvoiceDetails>
         <CPO>P0103940-1</CPO>
         <PNR>024147-000</PNR>
         <INV>INVAW01</INV>
         <INQ UNT="EA">1</INQ>
         <UNP ICR="USD" UNT="EA">100.00</UNP>
         <IND>2017-06-27</IND>
         <INA ICR="USD" UNT="EA">100.00</INA>
         <ITC>D</ITC>
         <PAY ICR="USD" UNT="EA">100.00</PAY>
         <SHT>DFW</SHT>
         <BOL>BOL9922</BOL>
      </InvoiceDetails>
   </ATA_SparesInvoice>
```

ATA_InvoiceMessageAck.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <ATA_InvoiceMessageAck version="1.12">
      <InvoiceAckHeader>
         <CIC>CMP</CIC>
         <SPL>CAGE1</SPL>
         <MessageReceiver>superuser</MessageReceiver>
      </InvoiceAckHeader>
      <InvoiceAckDetails>
         <CPO>P0103940-1</CPO>
         <PNR>024147-000</PNR>
         <INV>INVAW01</INV>
         <ADS>1</ADS>
      </InvoiceAckDetails>
   </ATA_InvoiceMessageAck>
```

B

Message element descriptions

The XML messages that are used to communicate information between Maintenix, AeroBuy, and vendor systems are defined by the Spec 2000 E-Business Specification for Materiels Management Guide, Revision 2014.1 (20th Revision).

When configuring the API or troubleshooting messages, you may need to know what information is sent in an element, whether it's mandatory, its data type, and where it maps to in the Maintenix user interface.

ATA_SparesQuoteRequest

The ATA_SparesQuoteRequest (R2014.1) message is sent when RFQs are issued, updated or canceled.

The ATA_SparesQuoteRequest message corresponds to the *RFQ placement scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the ATA_SparesQuoteRequest elements.

Version

Mandatory	Yes
Description	Version of the XSD that is supported. The default setting is 1.01
Data type	Decimal
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code
Data type	String (5)
Maps to Maintenix UI	RFQ Details > Details tab > Spec 2000 Customer Code

SPL

Mandatory	Yes
Description	Vendor code.
Data type	String (5)
Maps to Maintenix UI	RFQ Details > Vendor Quotes tab > Vendor

MessageSender

Mandatory	No
Description	Name of Maintenix user that sends the message.
Data type	String (20)
Maps to Maintenix UI	RFQ Details > Details tab > Purchasing Contact

MessageReceiver

Mandatory	No
Description	The user receiving the message in AeroBuy.
Data type	String (20)
Maps to Maintenix UI	N/A

QTR

Mandatory	Yes
Description	RFQ number.

Data type	String (11)
Maps to Maintenix UI	RFQ Details > RFQ Number

PNR

Mandatory	Yes
Description	Spec 2000 part number.
Data type	String (32)
Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Line Description

QTQ

Mandatory	Yes
Description	Number of parts requested.
Data type	String (5)
Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Quantity

QTQ.UNT

Mandatory	No
Description	Unit of measure code.
Data type	String (2)
Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Quantity

PRI

Mandatory	No
Description	Priority code.
Data type	String (3)

Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Priority

SSD

Mandatory	No
Description	Specified shipping date.
Data type	Date [YYYY-MM-DD]
Maps to Maintenix UI	RFQ Details > Details tab > Needed By

MFR

Mandatory	No
Description	Manufacturer code.
Data type	String (5)
Maps to Maintenix UI	N/A

TAD

Mandatory	No
Description	Traceability data indicator. A value of YES indicates that all documentation certifying origin and condition is required with the purchase of this part.
Data type	String (3)
Maps to Maintenix UI	N/A

CND

Mandatory	No
Description	Condition code.
Data type	String (2)

Maps to Maintenix UI	N/A

REM

Mandatory	No
Description	Remarks text.
Data type	String (1000)
Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Vendor Note

ATA_SparesQuoteInterim

The ATA_SparesQuoteInterim (R2014.1) message may be sent from AeroXchange by the vendor in acknowledgement of an RFQ.

The ATA_SparesQuoteInterim message corresponds to the *RFQ acknowledgement scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the ATA_SparesQuoteInterim elements.

Version

Mandatory	Yes
Description	Version of the XSD that is supported. The default setting is 1.01
Data type	Decimal
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)

Maps to Maintenix UI	RFQ Details > Details tab > Spec 2000 Customer Code

SPL

Mandatory	Yes
Description	Supplier code.
Data type	String (5)
Maps to Maintenix UI	RFQ Details > Vendor Quotes tab > Vendor

MessageSender

Mandatory	No
Description	The user that sent the message from AeroBuy
Data type	String (20)
Maps to Maintenix UI	RFQ Details > Details tab > Purchasing Contact

MessageReceiver

Mandatory	No
Description	The user the received the message in Maintenix.
Data type	String (20)
Maps to Maintenix UI	N/A

QTR

Mandatory	Yes
Description	RFQ number.
Data type	String (11)
Maps to Maintenix UI	RFQ Details > RFQ Number

PNR

Mandatory	Yes
Description	Spec 2000 part number.
Data type	String (32)
Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Line Description

MFR

Mandatory	No
Description	Manufacturer code.
Data type	String (5)
Maps to Maintenix UI	N/A

REM

Mandatory	No
Description	Vendor remarks.
Data type	String (1000)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Vendor Note

ATA_SparesQuoteFinal

The ATA_SparesQuoteFinal (R2014.1) message is sent from AeroXchange by the vendor as a response to an RFQ.

The ATA_SparesQuoteFinal message corresponds to the *Quote receipt scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the ATA_SparesQuoteFinal elements.

Version

Mandatory	Yes
Description	Version of the XSD that is supported. The default setting is 1.01
Data type	Decimal
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)
Maps to Maintenix UI	RFQ Details > Details tab > Spec 2000 Customer Code

SPL

Mandatory	Yes
Description	Supplier code.
Data type	String (5)
Maps to Maintenix UI	RFQ Details > Vendor Quotes tab > Vendor

MessageSender

Mandatory	No
Description	The user that sent the quote from AeroBuy.
Data type	String (20)
Maps to Maintenix UI	RFQ Details > Details tab > Purchasing Contact

MessageReceiver

Mandatory	No
Description	The user that received the quote in Maintenix.

Data type	String (20)
Maps to Maintenix UI	N/A

QTR

Mandatory	Yes
Description	RFQ number.
Data type	String (11)
Maps to Maintenix UI	RFQ Details > RFQ Number

IPN

Mandatory	Yes
Description	The part number originally requested by the customer.
Data type	String (32)
Maps to Maintenix UI	RFQ Details > RFQ Lines tab > Line Description

RPN

Mandatory	Yes
Description	Replacement part number. If there is no replacement part, this field should duplicate IPN.
Data type	String (32)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Quoted Part

INC

Mandatory	No*
Description	Interchangeability code between the old part (IPN) and the replacement part (RPN). *Required if RPN is different than IPN

Data type	String (1)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Vendor Note

QTQ

Mandatory	Yes
Description	Quotation quantity.
Data type	Decimal
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Quote Qty

QTQ.UNT

Mandatory	Yes
Description	Quantity unit of measure.
Data type	String (2)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Quote Qty

KWD

Mandatory	Yes
Description	Keyword name. The key, single-word name of the subject part or material item.
Data type	String (8)
Maps to Maintenix UI	N/A

UNP

Mandatory	Yes
Description	Unit price amount.
Data type	Decimal

Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Unit Price

UNP.ICR

Mandatory	Yes
Description	International currency code for the unit price (UNP).
Data type	String (3)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Unit Price

UNP.UNT

Mandatory	No
Description	Unit price unit of measure. This information is already captured in the RFQ.
Data type	String (2)
Maps to Maintenix UI	N/A

PFT

Mandatory	Yes
Description	Number of days the price held firm, starting on the Quotation Date (QTD).
Data type	Integer (3)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Effective To

LTM

Mandatory	Yes
Description	Lead time in days.
Data type	Integer (3)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Lead Time

QTN

Mandatory	Yes
Description	Quotation number.
Data type	String (15)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Quote Number

QTD

Mandatory	Yes
Description	Quotation date.
Data type	Date [YYYY-MM-DD]
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Quote Date

SPQ

Mandatory	No
Description	Standard package quantity.
Data type	Decimal
Maps to Maintenix UI	N/A

SPQ.UNT

Mandatory	No
Description	Unit of measure of standard package quantity.
Data type	String (2)
Maps to Maintenix UI	N/A

MSQ

Mandatory	No

Description	Minimum sales quantity.
Data type	Decimal
Maps to Maintenix UI	N/A

MSQ.UNT

Mandatory	No
Description	Unit of measure of minimum sales quantity.
Data type	String (2)
Maps to Maintenix UI	N/A

PBQ

Mandatory	No
Description	Price break quantity. If provided, it is the minimum number or amount of a item that must be ordered for a price break to apply.
Data type	Decimal
Maps to Maintenix UI	N/A

PBQ.UNT

Mandatory	No
Description	Unit of measure of price break quantity.
Data type	Decimal
Maps to Maintenix UI	N/A

PBP

Mandatory	No
Description	Unit price for a specific price break quantity range.

Data type	Decimal
Maps to Maintenix UI	N/A

PBP.ICR

Mandatory	No
Description	Price break price international currency code.
Data type	String (3)
Maps to Maintenix UI	N/A

PBP.UNT

Mandatory	No
Description	Unit of measure of price break price.
Data type	String (2)
Maps to Maintenix UI	N/A

CAT

Mandatory	No
Description	Category I container price amount. The discounted unit selling price for one each of the Category I Container available for transport and storage of the subject part number.
Data type	Decimal
Maps to Maintenix UI	N/A

CAT.ICR

Mandatory	No
Description	Category I container price international currency code.
Data type	String (3)

Maps to Maintenix UI	N/A

CAT.UNT

Mandatory	No
Description	Category I container price unit of measurement.
Data type	String (2)
Maps to Maintenix UI	N/A

MFR

Mandatory	No
Description	Part manufacturer.
Data type	String (5)
Maps to Maintenix UI	N/A

RFC

Mandatory	No*
Description	Reason for part number change. Specifies the technical authority or reason for the addition, deletion, or change to a part 2000, number. *This element is mandatory if RPN is different than IPN
Data type	String (12)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Vendor Note

DIS

Mandatory	No
Description	Discount percentage.
Data type	Decimal

Maps to Maintenix UI	N/A

TAD

Mandatory	No
Description	Traceability data indicator. A value of YES indicates that all documentation certifying origin and condition is required with the purchase of this part.
Data type	String (3)
Maps to Maintenix UI	N/A

UST

Mandatory	No
Description	Universal serial tracking number. Consists of the combination of the CAGE Code or NCAGE Code of the Supplier of data (SPL) for an in#service part, plus the Unique Component Identification Number (UCN) that will be a unique number within that CAGE or NCAGE Code.
Data type	String (20)
Maps to Maintenix UI	N/A

CRP

Mandatory	No
Description	Document reference indicator. A URL that identifies a document source on a particular web site.
Data type	String (99)
Maps to Maintenix UI	N/A

CND

Mandatory	No
Description	Condition code.
Data type	String (2)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Condition

SOR

Mandatory	No
Description	Part source code. Provides information in code form about the origin of the part.
Data type	String (3)
Maps to Maintenix UI	N/A

SRN

Mandatory	No
Description	Source name. Provides the name, a CAGE Code or NCAGE Code of a company or agency which issued the original source certification for a part.
Data type	String (30)
Maps to Maintenix UI	N/A

CES

Mandatory	No
Description	Certificate source. The identification of the company or agency which issued the certification of origin for a part. Source identification may consist of a company or agency name or a company CAGE Code or NCAGE Code.

Data type	String (30)
Maps to Maintenix UI	N/A

CRD

Mandatory	No
Description	Certificate date. Specifies the date on which a Part Origin Certificate was issued for a specific part.
Data type	Date [YYYY-MM-DD]
Maps to Maintenix UI	N/A

SER

Mandatory	No
Description	Part serial number.
Data type	String (30)
Maps to Maintenix UI	N/A

WON

Mandatory	No
Description	Work order number.
Data type	String (20)
Maps to Maintenix UI	N/A

DMF

Mandatory	No
Description	Manufacturer date.
Data type	Date [YYYY-MM-DD]

Maps to Maintenix UI	N/A

TTM.OTT

Mandatory	No
Description	Total accumulated operating time.
Data type	Integer (6)
Maps to Maintenix UI	N/A

TTM.OPC

Mandatory	No
Description	Total accumulated operating cycle count.
Data type	Integer (6)
Maps to Maintenix UI	N/A

TTM.TRF

Mandatory	No
Description	Time/cycle reference code. Designates the reference point for total accumulated operating time (TTM.OTT) or cycles (TTM.OPC).
Data type	String (1)
Maps to Maintenix UI	N/A

EXD

Mandatory	No
Description	Shelf life expiration date.
Data type	Date [YYYY-MM-DD]

Maps to Maintenix UI	N/A

PAK

Mandatory	No
Description	OEM original packaging code. `Y` or `N`. (Yes or No)
Data type	String (1)
Maps to Maintenix UI	N/A

REM

Mandatory	No
Description	Vendor remarks.
Data type	String (1000)
Maps to Maintenix UI	RFQ Details > Quote Comparison tab > Vendor Note

ATA_SparesOrder

The ATA_SparesOrder (R2014.1) message is sent when purchase orders are placed, updated, canceled, or when an order exception is rejected by the buyer. The message is composed of specific elements.

The ATA_SparesOrder message corresponds to the *Purchase order placement and update scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the ATA_SparesOrder elements.

Version

Mandatory	Yes
Description	Version of the XSD that is supported. The default setting is 1.01
Data type	Decimal

Maps to Maintenix UI	N/A

MessageSender

Mandatory	No
Description	Name of Maintenix user that sends the message.
Data type	String (100)
Maps to Maintenix UI	This will be populated with the username of the account from which the purchase order is being published from.

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)
Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Customer Code

SPL

Mandatory	Yes
Description	Supplier code.
Data type	String (5)
Maps to Maintenix UI	PO Details > Order Information area > Vendor

TNC.TNC_Order

Mandatory	Yes
Description	Order transaction code. If a priority is AOG then the value is 4 - Expedite Order. Otherwise, the value is 3 - Normal Order.
Data type	String (1)

Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Priority

TNC.TNC_Item

Mandatory	Yes
Description	Value: 4 - Replenishment stock item.
Data type	String (1)
Maps to Maintenix UI	N/A

TNC.TNC_Action

Mandatory	Yes
Description	Communicates the purpose of the message in terms of the action executed in the Maintenix user interface. • 1 - Routine Action (create purchase order line in Maintenix). • 2 - Acceptance of a quotation (N/A in Maintenix). • 3 - Alteration to an order (any change in Maintenix not captured in 1,6, or 8). • 6 - Cancel or decrease an order QTO (cancel purchase order line or decrease quantity in Maintenix). • 8 - Increase order QTO (increase purchase order line quantity in Maintenix). • 9 - Request partial shipment quantity of an open order (N/A in Maintenix).
Data type	String (1)
Maps to Maintenix UI	N/A

CPO

Mandatory	Yes
Description	Customer order number. Describes the order number and the line number concatenated with a hyphen (*order_number*-*line_number*).
Data type	String (32)

Maps to Maintenix UI	PO Details page > Order Information area > Order Number — PO Details page > Order Lines tab > Line Number

PNR

Mandatory	Yes
Description	Spec 2000 part number.
Data type	String (32)
Maps to Maintenix UI	Part Details page > Part area > OEM Part Number

UNP

Mandatory	Yes
Description	Unit price amount.
Data type	Decimal (12,2)
Maps to Maintenix UI	PO Details page > Order Lines tab > Price column

UNP.ICR

Mandatory	Yes
Description	International currency code. Note: USD is the only currency code configured in Maintenix for purchasing.
Data type	String (3)
Maps to Maintenix UI	PO Details page > Order Lines tab > Price column

UNP.UNT

Mandatory	No
Description	Unit of measure code.
Data type	String (2)

Maps to Maintenix UI	PO Details page > Order Lines tab > Qty column

QTO

Mandatory	Yes
Description	Order quantity. The QTO always conveys the total quantity of the PO. While the ICQ and the DCQ represent the delta for increase and decrease respectively. For quantity changes, QTO contains the change in the quantity. For example, if the initial PO was for 10 and you want to increase the quantity to 25, the QTO contains an element called ICQ which equals 15 and the QTO=25. The <TNC_Action> contains 8 on the change order. Conversely, if the initial PO was for 10 and you want to decrease the quantity, the QTO also contains an element called DCQ which equals 5 and the QTO=5.
Data type	Decimal (5,0)
Maps to Maintenix UI	PO Details page > Order Lines tab > Qty column

QTO.UNT

Mandatory	No
Description	Unit of measure code.
Data type	String (2)
Maps to Maintenix UI	PO Details page > Order Lines tab > Qty column

SSD

Mandatory	Yes
Description	Specified shipping date Format: YYYY-MM-DD
Data type	Date

Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Promised By

MFR

Mandatory	Yes
Description	Manufacturer code.
Data type	String (5)
Maps to Maintenix UI	Manufacturer Details page > Details > Manufacturer Code

SSM.ShippingMethodCode

Mandatory	Yes if SSM is provided.
Description	Specifies the method to use for transporting a specific shipment from the vendor's facility.
Data type	String (2)
Maps to Maintenix UI	PO Details page > Details tab > Shipping and Terms area > Transportation Type

SSM.ShippingPaymentMethodCode

Mandatory	Yes if SSM is provided.
Description	Specifies the method for paying the transportation costs.
Data type	String (1)
Maps to Maintenix UI	N/A

SHT

Mandatory	No
Description	Ship To code. The Ship To Code shows in the user interface when a SPEC 2000 vendor is selected for an order.
Data type	String (5)

Maps to Maintenix UI	Order Details page > Details tab > Shipping and Terms area > Ship To Code

PRI

Mandatory	No
Description	Priority code. The Maintenix SPEC2000_PO_PRIORITY_MAP configuration parameter must be used to map Maintenix priorities to SPEC 2000 priorities.
Data type	String (3)
Maps to Maintenix UI	Order Details page > Details tab > Identification area > Priority

ACN

Mandatory	Yes if PRI value corresponds to AOG. Otherwise optional.
Description	Aircraft registration number. The three to five most significant characters of the registration number. The number is obtained from the link between the PO and the part request. The request is linked to the task which is associated with the Aircraft.
Data type	String (3-5)
Maps to Maintenix UI	Inventory Details page > Details tab > Aircraft Information area > Registration Code

REM

Mandatory	No
Description	Remarks text
Data type	String (100)
Maps to Maintenix UI	PO Details page > Details tab > Note to Vendor area OR PO Details page > Order Lines tab > Vendor Note column

ATA_SparesMessageAck

Maintenix receives the ATA_SparesMessageAck (R2007.1) order acknowledgment message after AeroBuy or the vendor receive a purchase order. The message is composed of specific elements.

There are Preferences in AeroBuy for having the Acknowledgement sent automatically by AeroBuy or manually by the vendor. Some customers may want vendors to send the message manually so they know someone has seen it at the vendor location.

The ATA_SparesMessageAck message corresponds to the *Purchase order acknowledgment scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the elements in the ATA_SparesMessageAck message.

Version

Mandatory	Yes
Description	Version of the XSD that is supported. The default setting is 1.00
Data type	Decimal
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)
Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Customer Code

SPL

Mandatory	Yes
Description	Supplier Code.
Data type	String (5)

Maps to Maintenix UI	PO Details page > Details tab > Order Information area > Vendor

MessageTypeIndicator

Mandatory	Yes
Description	Contains the string 'Order'. Represents that the acknowledgment is for a purchse order.
Data type	String (5)
Maps to Maintenix UI	N/A

CPO

Mandatory	Yes
Description	Customer order number. Describes the order number and the line number concatenated with a hyphen (*order_number*-*line_number*).
Data type	String (32)
Maps to Maintenix UI	PO Details page > Details tab > Order Information area > Order Number - PO Details page > Order Lines tab, > Line No column

MessageReceiver

Mandatory	No
Description	Message receiver. Format: <Message Receiver> + '\n' + <REM>
Data type	String (20)
Maps to Maintenix UI	The purchase order message sender receives the acknowledgement message.

ATA_SparesOrderExc

The ATA_SparesOrderExc (R2014.1) order exception message is received when vendors communicate changes to purchase orders such as a part number, scheduled date, unit of measure, unit price, remarks, or quantity. The message is composed of specific elements.

The ATA_SparesOrderExc message corresponds to the *Order exception scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

Version

Mandatory	Yes
Description	Version of the XSD that is supported. The default setting is 1.00
Data type	Decimal
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)
Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Customer Code

SPL

Mandatory	Yes
Description	Supplier code.
Data type	String (5)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > Vendor column

CPO

Mandatory	Yes

Description	Customer order number. AeroBuy treats each line in an order as a separate order, so on an incoming exception, that number is created based on the original order number and the line number concatenated with a hyphen (*order_number-line_number*).
Data type	String (32)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > PO column The CPO here is broken down into two columns, there is a PO Column and a Line Column on the Order Exceptions tab

PNR

Mandatory	Yes
Description	Spec 2000 part number for part that was requested in initial order.
Data type	String (32)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > PartNumber column (number in regular font)

ReplacingPartData.RPN

Mandatory	No
Description	The part number the vendor proposes to supply in place of the part number that was requested on the order.
Data type	String (32)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > PartNumber column, number in bold font

ReplacingPartData.INC

Mandatory	No
Description	Interchangeability code
Data type	String (1)
Maps to Maintenix UI	N/A

UNP

Mandatory	Yes
Description	Unit price amount.
Data type	Decimal (12,2)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > Unit Price column

UNP.ICR

Mandatory	Yes
Description	International currency code. Note: USD is the only currency code configured in Maintenix for purchasing.
Data type	String (3)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > Unit Price column

UNP.UNT

Mandatory	No
Description	Unit of measure code.
Data type	String (2)
Maps to Maintenix UI	N/A

PDP

Mandatory	No
Description	Procurement discount percentage.
Data type	Decimal (4,2)
Maps to Maintenix UI	N/A

PKG

Mandatory	No
Description	Packaging code
Data type	String (1)
Maps to Maintenix UI	N/A

VRN

Mandatory	No
Description	VAT registration number
Data type	String (16)
Maps to Maintenix UI	N/A

MFR

Mandatory	No
Description	Manufacturer code. If the MFR is not present it is because it is identical to the supplier code. MFR is only required on an Order Exception message when the vendor suggests a replacement part.
Data type	String (5)
Maps to Maintenix UI	N/A

QuantityAdjustmentDetails.ICQ

Mandatory	No
Description	Increase quantity. The delta between the quantity requested by the buyer and the quantity the seller proposes to supply.
Data type	Integer
Maps to Maintenix UI	N/A

QuantityAdjustmentDetails.ICQ.UNT

Mandatory	No
Description	Unit of measure code for the increased quantity.
Data type	String(2)
Maps to Maintenix UI	N/A

QuantityAdjustmentDetails.DCQ

Mandatory	No
Description	Decrease quantity. The delta between the quantity requested by the buyer and the quantity the seller proposes to supply.
Data type	Integer
Maps to Maintenix UI	N/A

QuantityAdjustmentDetails.DCQ.UNT

Mandatory	No
Description	Unit of measure code for the decreased quantity. Unit of measure code for the decreased quantity.
Data type	String(2)
Maps to Maintenix UI	N/A

QuantityAdjustmentDetails.QTO

Mandatory	No
Description	Total quantity.
Data type	Integer
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > Quantity column

QuantityAdjustmentDetails.QTO.UNT

Mandatory	No
Description	Unit of measure code
Data type	String(2)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > Quantity column

ScheduleDetails.QTO

Mandatory	Mandatory if ScheduledDetails is provided.
Description	Total quantity.
Data type	Integer
Maps to Maintenix UI	N/A

ScheduleDetails.QTO.UNT

Mandatory	Mandatory if ScheduledDetails is provided.
Description	Unit of measure code.
Data type	String(2)
Maps to Maintenix UI	N/A

ScheduleDetails.QTO.SCD

Mandatory	Mandatory if ScheduledDetails is provided (choice of SCD or ODI).
Description	Scheduled shipping date.
Data type	Date
Maps to Maintenix UI	N/A

ScheduleDetails.QTOODI

Mandatory	Mandatory if ScheduledDetails is provided (choice of SCD or ODI).

Description	Open date indicator - not considered in Maintenix logic.
Data type	Boolean
Maps to Maintenix UI	N/A

REM

Mandatory	No
Description	Remarks.
Data type	String (1000)
Maps to Maintenix UI	Spec 2000 PO Exception Search page > Order Exceptions tab > Remarks column

ATA_SparesSupplierShipNotice

The ATA_SparesSupplierShipNotice (R2014.1) message is received when vendors communicate information about shipments. The message is composed of specific elements.

The ATA_SparesSupplierShipNotice corresponds to the *Shipment advisories scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)
Maps to Maintenix UI	PO Details > Details > Order Details area > Customer Code

SPL

Mandatory	Yes
Description	Supplier code.
Data type	String (5)

Maps to Maintenix UI	PO Details > Order Information area > Vendor

CPO

Mandatory	Yes
Description	Customer order number. Describes the order number and the line number concatenated with a hyphen (*order_number-line_number*).
Data type	String (32)
Maps to Maintenix UI	PO Details > Order Information area > Order Number - PO Details > Order Lines tab > Line Number column

PNR

Mandatory	Yes
Description	Spec 2000 part number. (Checked against the part number on the purchase order.)
Data type	String (32)
Maps to Maintenix UI	PO Details page > Order Lines tab > Line Description column

SHQ

Mandatory	Yes
Description	Shipment quantity.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > Shipment Lines tab> Expected Qty column

SHQ.UNT

Mandatory	Yes
Description	Unit of measure for the shipment quantity.
Data type	String (2)

Maps to Maintenix UI	Shipment Details page > Shipment Lines tab > Expected Qty column

SHT

Mandatory	Yes
Description	Ship To code.
Data type	String (5)
Maps to Maintenix UI	Shipment Details page > Details tab > Identification area > Ship To

SHD

Mandatory	Yes
Description	Shipped date.
Data type	Date [YYYY-MM-DD]
Maps to Maintenix UI	Shipment Details page > Details tab > Details area > Shipment Date

PSN

Mandatory	No
Description	Packing sheet number.
Data type	String (15)
Maps to Maintenix UI	History noted added: Shipment Details page > History tab > Note

ShipNoticeShipment.BOL

Mandatory	Yes
Description	Bill of lading number.
Data type	String (24)
Maps to Maintenix UI	Shipment Details page> Details tab > Details area > Waybill Number

ShipNoticeShipment.BLS

Mandatory	No
Description	Bill of lading - container count. The total number of containers documented on the bill of lading for a shipment.
Data type	Decimal (3,0)
Maps to Maintenix UI	History noted added: Shipment Details page > History tab > Note

ShipNoticeShipment.BLS.UNT

Mandatory	Yes if ShipNoticeShipment.BLS is provided.
Description	Bill of lading - container count unit of measure.
Data type	String (2)
Maps to Maintenix UI	History noted added: Shipment Details page > History tab > Note

ShipNoticeShipment.BOL_Weight

Mandatory	No
Description	Bill of lading - weight.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page >Weight

ShipNoticeShipment.BOL_Weight.UNT

Mandatory	Yes if ShipNoticeShipment. BOL_Weight is provided.
Description	Bill of lading - weight unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.BOX

Mandatory	Yes if ShipNoticeShipment.Carton is provided.

Description	Master carton number.
Data type	String (10)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.BOX.UNT

Mandatory	Yes if ShipNoticeShipment.Carton.BOX is provided.
Description	Carton dimensions - unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.UNT

Mandatory	No
Description	Carton dimensions - unit of measure. Note: if the Maintenix engineering unit of measure is not set up to coincide with the SPEC2000 unit of measure, data mapping is required.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.LEN

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions is provided.
Description	Carton length.
Data type	Decimal
Maps to Maintenix UI	Shipment Details > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.LEN.UNT

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions.LEN is provided.
Description	Carton length - unit of measure.

Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.WID

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions is provided. Only for the first occurence of the Carton unbound sequence.
Description	Carton width.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.WID.UNT

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions.WID is provided.
Description	Carton width - unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.HGT

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions is provided. Only for the first occurrence of the Carton unbound sequence.
Description	Carton height.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.HGT.UNT

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions.HGT is provided.
Description	Carton height - unit of measure.
Data type	String (2)

Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipNoticeShipment.Carton.CartonDimensions.CartonWeight

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions is provided.
Description	Carton weight.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > History tab > Note A history note is added for the first occurrence of the Carton unbound sequence.

ShipNoticeShipment.Carton.CartonDimensions.CartonWeight.UNT

Mandatory	Yes if ShipNoticeShipment.Carton.CartonDimensions.CartongWeight is provided.
Description	Carton weight - unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note A history note is added for the first occurrence of the Carton unbound sequence.

ShipmentNoticeCarton.BOX

Mandatory	Yes if ShipmentNoticeCarton is provided.
Description	Master carton number.
Data type	String (10)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.BOX.UNT

Mandatory	Yes if ShipmentNoticeCarton.BOX is provided.
Description	Carton dimension - unit of measure.
Data type	String (2)

Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.UNT

Mandatory	No
Description	Carton dimensions - unit of measure. Note: If the Maintenix engineering unit of measure is not set up to coincide with SPEC2000 unit of measure, data mapping is required.
Data type	String (2)
Maps to Maintenix UI	Shipment Details > History tab > Note

ShipmentNoticeCarton.CartonDimensions.LEN

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions is provided.
Description	Carton length.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.LEN.UNT

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions.LEN is provided.
Description	Carton length - unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.WID

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions is provided.
Description	Carton width.
Data type	Decimal

Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.WID.UNT

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions.WID is provided.
Description	Carton width - unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.HGT

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions is provided.
Description	Carton height.
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.HGT.UNT

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions.HGT is provided.
Description	Carton height - unit of measure.
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.CartonWeight

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions is provided.
Description	Carton weight
Data type	Decimal
Maps to Maintenix UI	Shipment Details page > History tab > Note

ShipmentNoticeCarton.CartonDimensions.CartonWeight.UNT

Mandatory	Yes if ShipmentNoticeCarton.CartonDimensions.CartongWeight is provided.
Description	Carton weight - unit of measure
Data type	String (2)
Maps to Maintenix UI	Shipment Details page > History tab > Note

PRI

Mandatory	No
Description	Priority code. Note: Maintenix configuration parameter SPEC2000_PO_PRIORITY_MAP must be used to map Maintenix priorities to SPEC 2000 priorities.
Data type	String (3)
Maps to Maintenix UI	Shipment Details page > Details tab > Identification area > Priority

CAR

Mandatory	No
Description	Carrier name.
Data type	String (30)
Maps to Maintenix UI	Shipment Details page > Details tab > Details area > Carrier Name

SHM.ShippingMethodCode

Mandatory	Yes if SHM is provided.
Description	Actual shipping method code.
Data type	String (2)

Maps to Maintenix UI	Shipment Details > History tab > Note

SHM.ShippingPaymentMethodCode

Mandatory	Yes if SHM is provided.
Description	Payment method code.
Data type	String (1)
Maps to Maintenix UI	Information is added as a history note

SHF

Mandatory	No
Description	Shipped From location code
Data type	String (3)
Maps to Maintenix UI	Information is added as a history note

MFR

Mandatory	No
Description	Manufacturer code. If the MFR is not present it is because it is identical to the supplier code.
Data type	String (5)
Maps to Maintenix UI	Unused by Maintenix

ICC

Mandatory	No
Description	International commodity code.
Data type	String (6)
Maps to Maintenix UI	Information is added as a history note

SER

Mandatory	No
Description	Part serial number.
Data type	String (3)
Maps to Maintenix UI	Shipment Details > History tab > Note

REM

Mandatory	No
Description	Remarks text.
Data type	String (1000)
Maps to Maintenix UI	Information is added as a history note

ATA_SparesInvoice

Maintenix receives the ATA_SparesInvoice (R2014.1) message when vendors send invoices and invoice updates from the AeroBuy portal, AeroBuy Smartmail, or from their own system through Electronic Data Interchange (EDI). The message is composed of specific elements.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the ATA_SparesInvoice elements.

Version

Mandatory	Yes
Description	Version of the XSD supported. The default setting is 1.01
Data type	Decimal
Maps to Maintenix UI	N/A

MessageSender

Mandatory	No

Description	Name of the vendor user that sends message.
Data type	String (20)
Maps to Maintenix UI	This is populated with the name of the vendor's agent who is invoicing against a Maintenix purchase order.

MessageReceiver

Mandatory	No
Description	Name of the Maintenix user who is the contact for the vendor.
Data type	String (20)
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code that identifies the AeroXchange customer.
Data type	String (3-5)
Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Customer Code

SPL

Mandatory	Yes
Description	Supplier code that identifies the vendor.
Data type	String (5)
Maps to Maintenix UI	PO Details > Order Information area > Vendor

CPO

Mandatory	Yes
Description	Customer order number. Describes the order number and the line number concatenated with a hyphen (*`order_number-line_number`*).
Data type	String (11)

Maps to Maintenix UI	PO Details page > Order Information area > Order Number — PO Details page > Order Lines tab > Line Number

PNR

Mandatory	Yes
Description	Spec 2000 part number. This number must match the part number in the purchase order. It cannot be an alternate part number.
Data type	String (32)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Line Description

INV

Mandatory	Yes
Description	Invoice number. When communicating through AeroBuy, each line of an Maintenix purchase order is a separate PO in AeroBuy. Invoices correspond, one for one, with each separate PO.
Data type	String (25)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Information area > PO Invoice Number

INQ

Mandatory	Yes
Description	Invoice quantity.
Data type	Integer (5)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Qty

INQ.UNT

Mandatory	Yes
Description	Unit of measure code.
Data type	String (2)

Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Qty

UNP

Mandatory	Yes
Description	Unit price amount.
Data type	Decimal (12, 2)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Unit Price

UNP.ICR

Mandatory	Yes
Description	International currency code. Must match the currency code on the original purchase order.
Data type	String (3)
Maps to Maintenix UI	N/A

UNP.UNT

Mandatory	Yes
Description	Unit price unit of measure code.
Data type	String (2)
Maps to Maintenix UI	N/A

IND

Mandatory	Yes
Description	Invoice date.
Data type	Date
Maps to Maintenix UI	PO Invoice Details page > Details tab, Details area > PO Invoice Date

INA

Mandatory	Yes, but not used by the Maintenix AeroBuy integration.
Description	Invoice net amount.
Data type	Decimal (13,2)
Maps to Maintenix UI	N/A

INA.ICR

Mandatory	Yes, but not used by the Maintenix AeroBuy integration.
Description	International currency code.
Data type	String (3)
Maps to Maintenix UI	N/A

INA.UNT

Mandatory	Yes, but not used by the Maintenix AeroBuy integration.
Description	Unit of measure code.
Data type	String (2)
Maps to Maintenix UI	N/A

ITC

Mandatory	Yes
Description	Invoice type code. There are two possible values: either C for credit, or D for debit.
Data type	String (1)
Maps to Maintenix UI	N/A

PAY

Mandatory	Yes, but not used by the Maintenix AeroBuy integration.

Description	Payable amount.
Data type	Decimal (13, 2)
Maps to Maintenix UI	N/A

PAY.ICR

Mandatory	Yes, but not used by the Maintenix AeroBuy integration.
Description	International currency code.
Data type	String (3)
Maps to Maintenix UI	N/A

PAY.UNT

Mandatory	Yes, but not used by the Maintenix AeroBuy integration.
Description	Unit of measure.
Data type	String (2)
Maps to Maintenix UI	N/A

SHT

Mandatory	Yes
Description	Ship To Code.
Data type	String (5)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Shipment Related Details area > Ship-To Code

PSN

Mandatory	Yes
Description	Packing sheet number.
Data type	String (15)

Maps to Maintenix UI	PO Invoice Details page > Details tab > Shipment Related Details area > Packing Sheet Number

BOL

Mandatory	No
Description	Bill of lading number. Waybill is another name for a bill of lading.
Data type	String (24)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Shipment Related Details area > Bill of Lading Number

CDT.CDC

Mandatory	No
Description	Cash discount percentage. (Vendors can offer multiple cash discount percentages and number of days combinations in the same message.)
Data type	Decimal (2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Cash Discount area > Cash Discount Maintenix shows only the highest discount percentage from the CDT.CDC element.

CDT.CDM

Mandatory	Yes, if cash discount percentage is provided.
Description	Cash discount terms. The number of days from the date of the invoice that the discount is valid for. Vendors can offer multiple cash discount percentages and number of days combinations.
Data type	Integer (3)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Cash Discount area > Discount Expiry Date Maintenix shows information from the CDT.CDM element that corresponds to the largest discount. The integer value is added to the vendor invoice date sent in the IND element to populate the expiry date.

OCH.OCC

Mandatory	No
Description	Other charge code.
Data type	String (2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Other Charges table

OCH.OCA

Mandatory	Yes if OCH.OCC is provided.
Description	Other charge amount.
Data type	Decimal (12, 2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Other Charges table

OCH.OCA.ICR

Mandatory	Yes if OCH.OCA is provided.
Description	International currency code.
Data type	String (3)
Maps to Maintenix UI	N/A

OCH.OCA.UNT

Mandatory	Yes if OCH.OCA is provided.
Description	Unit of measure.
Data type	String (2)
Maps to Maintenix UI	N/A

TaxAmountData.TAX

Mandatory	Yes if any TaxAmountData of TXL, TXS, TXC or VAT is provided.
Description	Total tax amount.

Data type	Decimal (12, 2)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Unit Price

TaxAmountData.TAX.ICR

Mandatory	Yes if TaxAmountData.TAX is provided.
Description	International currency code.
Data type	String (3)
Maps to Maintenix UI	N/A

TaxAmountData.TAX.UNT

Mandatory	No
Description	Unit of measure.
Data type	String (2)
Maps to Maintenix UI	N/A

TaxAmountData.TXL

Mandatory	No
Description	Local sales tax percentage.
Data type	Decimal (4, 2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Sales Tax table

TaxAmountData.TXS

Mandatory	No
Description	State sales tax percentage.
Data type	Decimal (4, 2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Sales Tax table

TaxAmountData.TXC

Mandatory	No
Description	County sales tax percentage.
Data type	Decimal (4, 2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Sales Tax table

TaxAmountData.VAT.VCC

Mandatory	No
Description	VAT category code.
Data type	String (1)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Value Added Taxes table

TaxAmountData.VAT.VAP

Mandatory	Yes, if TaxAmountData.VAT.VCC is provided.
Description	VAT percentage.
Data type	Decimal (4, 2)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Taxes and Other Charges area > Value Added Taxes table

SPD

Mandatory	No
Description	Invoice tax point date.
Data type	Date
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Tax Point Date

TAX

Mandatory	No
Description	Total tax amount.

Data type	Decimal (12, 2)
Maps to Maintenix UI	N/A

ADS

Mandatory	No, if the message is a new invoice, including new invoices that supersede existing invoices. See *Invoice updates* for more information. Yes, if the message is an update to an existing invoice.
Description	Adjustment sequence number. The first update to an invoice has an ADS value of 1. The value increments sequentially for subsequent updates to the same invoice.
Data type	String (1)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Miscellaneous area > Adjustment Sequence Number

RIN

Mandatory	No
Description	Reference invoice number. When a new invoice is created to supersede an existing invoice, RIN is used to identify the existing invoice.
Data type	String (25)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Miscellaneous area > Reference Invoice Number

HTC

Mandatory	No
Description	Harmonized tariff code.
Data type	String (15)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Miscellaneous area > Harmonized Tariff Code

REM

Mandatory	No
Description	Remarks text.
Data type	String (1000)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Note

ATA_InvoiceMessageAck

Maintenix sends an outbound ATA_InvoiceMessageAck message to AeroBuy communicating to the vendor that an invoice (or invoice update) was received. The message is composed of specific elements.

The ATA_InvoiceMessageAck message corresponds to the *Invoice acknowledgement scenario*.

Sample XML messages are provided in the documentation folder in the Maintenix installation location.

The following list describes the message elements:

Version

Mandatory	Yes
Description	Version of the XSD that is supported by the Maintenix AeroBuy integration.
Data type	Decimal (0,2)
Maps to Maintenix UI	N/A

CIC

Mandatory	Yes
Description	Customer identification code.
Data type	String (5)
Maps to Maintenix UI	PO Details page > Details tab > Order Details area > Customer Code

SPL

Mandatory	Yes
Description	Vendor code.
Data type	String (5)
Maps to Maintenix UI	PO Details > Order Information area > Vendor

MessageSender

Mandatory	No
Description	Message sender - Maintenix user.
Data type	String (20)
Maps to Maintenix UI	N/A

MessageReceiver

Mandatory	No
Description	Message receiver.
Data type	String (20)
Maps to Maintenix UI	N/A

CPO

Mandatory	Yes
Description	Customer order number. Identifies the order number and the line number concatenated with a hyphen. *<order_number>-<line_number>*
Data type	String (11)
Maps to Maintenix UI	PO Details page > Order Information area > Order Number — PO Details page > Order Lines tab > Line Number

PNR

Mandatory	Yes
Description	OEM Part Number. This number must match the part number in the purchase order.
Data type	String (32)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Lines tab > Line Description

INV

Mandatory	Yes
Description	Invoice number.
Data type	String (25)
Maps to Maintenix UI	PO Invoice Details page > PO Invoice Information area > PO Invoice Number

ADS

Mandatory	No, if the corresponding inbound invoice message did not contain an ADS element. Yes, if the corresponding inbound invoice message contained an ADS element.
Description	Adjustment sequence number. Used when invoices are updated.
Data type	String (1)
Maps to Maintenix UI	PO Invoice Details page > Details tab > Miscellaneous area > Adjustment Sequence Number

AEX_INVALIDS2K

Maintenix receives an AEX_INVALIDS2K (AeroBuy Spec 2000 v16) rejection message from AeroBuy if an outbound message from Maintenix was invalid (in format or construction). The message is composed of specific elements.

CmdCode

Mandatory	No

Description	Command code. Example: ATA_SparesOrder
Data type	String
Maps to Maintenix UI	Alerts page > Message

DocID

Mandatory	No
Description	Number that uniquely identifies the order. Format is `<PO Number>-<PO Line Number` Example: PO123454-1
Data type	String(11)
Maps to Maintenix UI	Alerts page > Message

USERAREA.VALUE

Mandatory	Yes
Description	Description of the error that caused the communication failure. Example: FIELD TOO LONG CPO 22 CHARS 11 MAX
Data type	String(4000)
Maps to Maintenix UI	Alerts page > Message

ABOUT IFS

IFS develops and delivers enterprise software for customers around the world who manufacture and distribute goods, maintain assets, and manage service-focused operations. The industry expertise of our people and solutions, together with commitment to our customers, has made us a recognized leader and the most recommended supplier in our sector. Our team of 3,500 employees supports more than 10,000 customers world-wide from a network of local offices and through our growing ecosystem of partners.

#forthechallengers

ifs.com

WHERE WE ARE

AMERICAS

+1 888 437 4968

ASIA PACIFIC

+65 63 33 33 00

EUROPE EAST

+48 22 577 45 00

EUROPE CENTRAL

+49 9131 77 340

UK & IRELAND

+44 1494 428 900

FRANCE, BENELUX AND IBERICA

+33 3 89 50 72 72

MIDDLE EAST AND AFRICA

+971 4390 0888

NORDICS

+46 13 460 4000

COPYRIGHT © 2026 INDUSTRIAL AND FINANCIAL SYSTEMS, IFS AB. IFS AND ALL IFS PRODUCTS AND SERVICES NAMES ARE TRADEMARKS OF IFS. ALL RIGHTS RESERVED. THIS DOCUMENT MAY CONTAIN STATEMENTS OF POSSIBLE FUTURE FUNCTIONALITY FOR IFS'S PRODUCTS AND TECHNOLOGY. SUCH STATEMENTS ARE FOR INFORMATION PURPOSES ONLY AND SHOULD NOT BE INTERPRETED AS ANY COMMITMENT OR REPRESENTATION. THE NAMES OF ACTUAL COMPANIES AND PRODUCTS MENTIONED HEREIN MAY BE THE TRADEMARKS OF THEIR RESPECTIVE OWNERS.