IFS Aviation Maintenance

Procurement API Guide

ICAM 26.2



Document Revision: 26.2-1

Publication Date: November 02, 2026

Contents

1 Introduction....7
About Maintenix Integrations....7
The Maintenix Procurement API....8

2 Setting up the Procurement API....10
License requirement....10
Permissions and authentication....11
Assign the Procurement integration role to a user....11
Give Procurement integration permission to a user role....12
Configuration parameters....12
Standard reference terms....13
Alerts....13
Activating alerts....13

3 API scenarios....14
Operational service scenarios....14
Sending a request for quote (RFQ) from Maintenix....14
Sending a purchase order from Maintenix....15
Requesting issued orders from Maintenix....15
Sending order exceptions to Maintenix....16
Sending shipment advisory to Maintenix....16
Creating an order invoice in Maintenix....17
Updating an order invoice in Maintenix....17
Updating a vendor's external key....17
Adding part prices in Maintenix....18
Updating part prices in Maintenix....18
Marking part prices obsolete in Maintenix....19
Notification service scenarios....19
Publishing vendors created in Maintenix....19
Publishing updates to a vendor's approval....20

4 Operational service messages....21
ATA_SparesQuoteRequest....21
Sample messages....22

Message triggers....22
Mapping elements to Maintenix GUI....22
ATA_SparesQuoteInterim....23
Sample messages....23
Message contents....23
Mandatory elements....23
Errors and exceptions....24
Mapping elements to Maintenix GUI....24
ATA_SparesQuoteFinal....25
Sample messages....25
Message contents....25
Mandatory elements....26
Errors and exceptions....27
Mapping elements to Maintenix GUI....27
getIssuedPurchaseOrders-1.0....28
Sample messages....28
Message contents....28
Mandatory elements....29
Errors and exceptions....29
Mapping elements to Maintenix GUI....29
S1CPOXMT-1.0....29
Sample messages....30
Message contents....30
Mapping elements to Maintenix GUI....31
S1CPOACK-1.0....32
Sample messages....32
Message contents....32
Mandatory elements....32
Errors and exceptions....33
Mapping elements to Maintenix....33
S1EXCXMT-1.0....34
Sample messages....34
Message contents....34
Mandatory elements....34
Errors and exceptions....35
Mapping elements to Maintenix....36
S1SPLSHP-1.0....37

Sample messages....37
Message contents....37
Mandatory elements....38
Errors and exceptions....39
Mapping elements to Maintenix....39
ATA_SparesInvoice-1.0....40
Sample messages....40
Message contents....40
Mandatory elements....41
Errors and exceptions....42
Mapping elements to Maintenix....42
ATA_InvoiceMessageAck-1.0....45
MxPartPrice-1.0....45
Sample messages....45
Message contents....46
Mandatory elements....46
Errors and exceptions....47
Mapping elements to Maintenix....47
MxPartPriceResponse-1.0....49
update_vendor_external_key-1.0....49
Sample messages....49
Message contents....49
Mandatory elements....49
Errors and exceptions....50
Mapping elements to Maintenix....50
create-vendor-request-1.0....50
Sample messages....51
Message contents....51
Mandatory elements....51
Errors and exceptions....52
Mapping elements to Maintenix....53
create-vendor-response-1.0....55
update-vendor-request-1.0....55
Message contents....55
Mandatory elements....55
Errors and exceptions....57
Mapping elements to Maintenix....57

update-vendor-response-1.0 ... 59
unapprove-vendor-request-1.0 ... 59
Message contents ... 59
Mandatory elements ... 60
Errors and exceptions ... 60
Mapping elements to Maintenix ... 60
unapprove-vendor-response-1.0 ... 61

5 Notification service messages ... 62
vendor-created-1.1 ... 62
Sample messages ... 62
Message triggers ... 63
Mapping elements to Maintenix GUI ... 63
vendor_approval_updated-1.0 ... 64
Sample messages ... 64
Message triggers ... 65
Mapping elements to Maintenix GUI ... 65

6 Integration framework ... 66
Operational and querying service messages ... 66
SOAP request message structure ... 68
SOAP response message structure ... 69
Notification service messages ... 71
Configuring Maintenix for the external system's web service ... 72
SOAP request message structure - notification service ... 74
SOAP response message structure - notifications ... 76
Setting up the IFS Maintenix Message Ordering Processor ... 76
Message streams ... 76
Adding new message streams ... 76
Configuring message order timing ... 77
Sequencing inbound messages ... 78
Managing integration messages in Maintenix ... 79
Viewing integration messages ... 79
Sending messages manually within Maintenix ... 81
Send message manually ... 82
Edit and resend an integration message ... 83
Understanding message errors ... 83

Archive integration messages..... 84
Logging..... 84

1

Introduction

About Maintenix Integrations

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 automation 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.

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 the successful receipt 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, Maintenix sends an outbound message containing specific information to an external system upon established events in Maintenix.

Note Not all APIs support all messaging scenarios.

Depending on the integration, there are different framework options.

Older adapter-type integrations can be configured to use a SOAP web service or JMS framework, with or without using the IFS Maintenix Integration Framework as a gateway in the integration architecture.

The IFS Maintenix Integration Framework is not required to use any integration, but it is an option for a number of APIs.

Some newer integrations are accessed by sending JSON requests to a publicly exposed HTTP endpoint.

The Maintenix Procurement API

Materials management is at the heart of each organization in the aviation industry. For commercial airlines, performing aircraft maintenance on time requires materials to be available in a serviceable condition.

There are steps in the process of acquiring new parts, and most require manual activities, which makes the whole process more time consuming. Delays can cause a significant impact on the capability of an airline to perform a maintenance check on time.

SPEC2000 Order Administration Module is a standard developed by ATA group to allow an automated, standardized, paperless communication between airlines and vendors. The purpose of the order administration module is to automate the placing of orders, order exception routines, on-line enquiries, and to determine order status and automated shipment advisories plus electronic invoicing. Some of the functionalities of the SPEC2000 order administration module are supported by the Maintenix procurement API.

Maintenix procurement API supports the following scenarios:

- Sending a request for quote (RFQ) to a vendor, and receiving RFQ responses from that vendor.
- Sending a purchase order to a vendor.
- Receiving a purchase order exception from the vendor.
- Receiving shipment information for an issued purchase order from the vendor.
- Receiving order invoices from vendors.
- Receiving part price information from vendors.

To enable SPEC2000 communication with an external system, each vendor must be flagged as **Spec 2000 Enabled** on the Vendor Details page. You must also select the types of SPEC2000 commands that are enabled for each vendor on the Communication tab of the Vendor Details page. The following table lists the SPEC2000 commands in Maintenix.

COMMAND	DESCRIPTION
S1QUOTES	If enabled, Maintenix can send RFQs to a SPEC2000 vendors automatically, and receive RFQ responses from vendors.
S1BOOKED	If enabled, Maintenix can send POs to a SPEC2000 vendors automatically when the order is issued.
S1ORDEXC	If enabled, vendors can send requests to change POs in Maintenix. This command should be enabled if S1BOOKED is enabled.
S1SHIPPD	If enabled, vendors can send shipment advisories to Maintenix for POs sent using the S1BOOKED command. This command should be enabled if S1BOOKED is enabled.
S1INVCE	If enabled, vendors can create and update invoices in Maintenix. This command should be enabled if S1BOOKED is enabled.

2

Setting up the Procurement API

Most messages in the Procurement API are sent using:

- A web service or JMS framework that integrates directly with Maintenix. See *Integration framework*

In addition to using one of the above integration options, if you are using the following set of messages, they are sent with a different web service:

- `create-vendor-request-1.0`
- `create-vendor-response-1.0`
- `unapprove-vendor-request-1.0`
- `unapprove-vendor-response-1.0`
- `update-vendor-request-1.0`
- `update-vendor-response-1.0`

For the create, unapprove, and update vendor messages above:

- The WSDL is available at: `http://<maintenix_server>:<http_port>/procurement/VendorService?wsdl`
- The request API Endpoint is `http://<maintenix_server>:<http_port>/procurement/VendorService`

Configuration for the above sub-set of messages is covered in the following topics on permissions, configuration parameters, standard reference terms, and alerts.

License requirement

To use the Procurement API, you must have the `PROCUREMENT_ADAPTER` feature name added to your Maintenix `license.xml` file. Contact your Maintenix representative if you have any questions about licensing.

Permissions and authentication

When Maintenix and PROCUREMENT_ADAPTER are installed, a default user called `prcintegration` with the PRCINTEG role is automatically created. The PRCINTEG role, gives the `prcintegration` user permission to authenticate and authorize inbound messages from the Procurement API to Maintenix. Any other user created for this purpose should be assigned the PRCINTEG role.

If you already use the Procurement API and you upgrade, you can use your current users and roles without any changes. You also have the option to use the Procurement role that is created with the installation.

The PRCINTEG role is assigned the following permissions:

- API_CREATE_VENDOR_REQUEST
- API_UPDATE_VENDOR_REQUEST
- API_UNAPPROVE_VENDOR_REQUEST

After installing the Procurement extension API, you can assign specific user roles permission to use the PRCINTEG role, or grant different user roles permission to use this API.

Assign the Procurement integration role to a user

You can grant specific users permission to the Procurement API.

Steps

1. Go to the **Role Search** page.
2. Enter PNCINTEG in the **Role Code** field and click **Search**.
3. On the **Role Details** page, **Users** tab, click **Assign User**.

> **Tip** If the user already appears on the **Role Details** page, select the user from the list and click **Assign User**. Other wise, follow the steps below.

4. On the **User Search** page, do any of the following:
   - Click **Search** to see a list of all users in the system.
   - Enter the complete or partial name of the user in any of the user name fields and click **Search**.
5. Select a user from the list.
6. Click **Assign User**.

Give Procurement integration permission to a user role

You can grant users, with a specific role, permission to use the Procurement API.

Steps

1. Go to **Role Search** page.
2. On the **Role Search** page, do any of the following:
   - Click **Search** to see a list of all roles in the system.
   - Enter the code in the **Role Code** field and click **Search**.
   - Enter the complete or partial name of the role in the **Name** field and click **Search**.
3. Select a role from the list.
4. On the **Role Details** page, click the **Permission** tab.
5. Enter the procurement permission names in the **Param Name** field and click **Search**.
6. Select the permission check boxes and click **Enable**.
7. Click **OK**.

Configuration parameters

The UTL_CONFIG_PARM table contains the parameters that configure the Procurement API functionality as described in the table below.

Procurement API configuration parameters

Parameter	Description
INCOMING_ADAPTER _XML_VALIDATION	This parameter controls the schema validation of incoming XML messages. By default, schema validation is enabled. To disable schema validation, set this configuration parameter to FALSE.
OUTGOING_ADAPTER _XML_VALIDATION	This parameter controls the schema validation of outgoing XML messages. By default, schema validation is enabled. To disable schema validation, set this configuration parameter to FALSE.

Standard reference terms

The following standard reference term is used by the procurement API to facilitate SPEC2000 integrations:

- **REF_SPEC2K_CUST:** Identifies the customer code, which is assigned by SPEC2000. Typically, there is one unique SPEC2000 customer code per company; however, it is possible for a holding company to have a separate SPEC2000 customer code for each company in the holding. Populate the REF_SPEC2K_CUST table with the applicable SPEC2000 customer codes for your organization. The SPEC2000 customer code is mandatory when creating RFQs and POs from the Maintenix UI, and is used in SPEC2000 procurement API integrations.

Alerts

You can configure the following role-based alerts to notify selected roles of events pertaining to procurement integrations:

- **Purchase Order Exception:** An exception has been raised against a purchase order.
- **No Matching Shipment Record for Shipment Advisory:** No shipment record exists in Maintenix for a shipment advisory received from an external system.
- **Shipment Advisory:** A shipment advisory has been received from an external system.

Activating alerts

To identify when an action fails, the Procurement API alerts must be activated.

Steps

1. Go to the **Alerts Setup** page.
2. Click the **Category** drop-down list and select "JOB".
3. Select the check box beside the Procurement API alert.
4. Click **Activate Alert**.

> **Tip** You can edit the roles assigned to alerts by selecting the check box beside the alert and clicking **Edit Role Assignments**. From this page you can select or clear roles as required.

3

API scenarios

This section describes the scenarios that are supported by the API.

Each of the supported scenarios contain the following sections:

- **Goal**: A short statement that defines the purpose of the scenario.
- **Summary**: A more detailed description of the scenario with additional context for the integration.
- **Preconditions**: A list of conditions that must be met before the scenario can be implemented.
- **Messaging**: A basic description of the messaging required for the scenario.
- **Result**: The outcome of the successful execution of the scenario.

Operational service scenarios

The Procurement API supports the following operational service scenarios.

Sending a request for quote (RFQ) from Maintenix

Goal: To send an RFQ from Maintenix to a vendor.

Summary: An RFQ is prepared in Maintenix. When the RFQ is sent within Maintenix, an RFQ outbound message is sent to SPEC 2000 enabled vendors. The outbound message can contain a maximum of 10 part numbers. If the vendor does not have a price available immediately, the vendor can respond with an interim response message. Once the vendor has a price to submit to Maintenix, the vendor sends a final quote message. An RFQ outbound message may contain up to ten part numbers, but for each part number, the vendor must send separate interim and final response messages.

Preconditions:

- The vendor is SPEC 2000 enabled.

- The vendor has the S1QUOTES command enabled.
- The RFQ is transitioned to SENT in Maintenix.

Messaging: Maintenix sends an outbound *ATA_SparesQuoteRequest* message. The external system returns an *S1CPOACK-1.0* or a *ATA_SparesQuoteFinal* inbound message.

Result: The RFQ message is sent to the vendor. When the vendor sends a response message, the details are recorded in Maintenix. Maintenix will also create a new part price for that vendor record that is used when you create a purchase order.

Sending a purchase order from Maintenix

Goal: To send a purchase order from Maintenix to a vendor.

Summary: An an order is prepared in Maintenix. When the order is issued within Maintenix, an outbound order message is sent to the SPEC 2000 enable vendors. A vendor can respond to acknowledge the receipt of the order, or a vendor can request changes to the order.

Preconditions:

- The vendor is SPEC 2000 enabled.
- The vendor has the S1BOOKED command enabled.
- The purchase order is transitioned to ISSUED in Maintenix.

Messaging: Maintenix sends an outbound *S1CPOXMT-1.0* message. The external system returns either an *S1CPOACK-1.0* inbound message to acknowledge the order, or an *S1EXCXMT-1.0* inbound message to request changes to the order.

Result: The order message is sent to the vendor.

Requesting issued orders from Maintenix

Goal: To request orders that have been issued in Maintenix.

Summary: An an order is prepared in Maintenix. When the order is issued within Maintenix, a vendor can request all issued orders. Maintenix then sends an outbound order message containing the details of all orders issued to that vendor. A vendor can respond to acknowledge the receipt of the order, or a vendor can request changes to the order.

Preconditions:

- The vendor is SPEC 2000 enabled.
- The vendor has the S1BOOKED command enabled.
- The purchase order is transitioned to ISSUED in Maintenix.

Messaging: The external system sends an inbound *getIssuedPurchaseOrders-1.0* message. Maintenix sends an outbound *S1CPOXMT-1.0* message. The external system returns either an

S1CPOACK-1.0 inbound message to acknowledge the order, or an *S1EXCXMT-1.0* inbound message to request changes to the order.

Result: The requested orders are sent to the vendor. If necessary, the vendor submits changes to the order, or the vendor acknowledges the order.

Sending order exceptions to Maintenix

Goal: To notify Maintenix of an exception to an order.

Summary: The vendor sends an exception to an order in Maintenix. An alert is sent to the purchasing manager, who may update the purchase order manually.

Preconditions:

- The vendor is SPEC 2000 enabled.
- The vendor has the S1ORDEXC command enabled.
- The purchase order is transitioned to ISSUED in Maintenix.

Messaging: The external system sends an inbound *S1EXCXMT-1.0* message.

Result: The order exception message is sent to Maintenix and the purchasing manager is alerted.

Sending shipment advisory to Maintenix

Goal: To update a shipment record in Maintenix.

Summary: When a SPEC 2000 enabled vendor has received and processed a purchase order from Maintenix, that vendor can send an inbound message to Maintenix to update the shipment records with detailed shipping information, and update the status of the shipment to IN-TRANS. You can also assign a **Shipment Advisory** alert to notify roles when Maintenix receives a shipment advisory message from an external system, or a No **Matching Shipment Record for Shipment Advisory** alert to notify roles when Maintenix receives a shipment advisory message from an external system that does not match an existing shipment in Maintenix.

Preconditions:

- The vendor is SPEC 2000 enabled.
- The vendor has the S1SHIPPD command enabled.
- The purchase order, order line number and part number identify an order that has been issued in Maintenix.

Messaging: The external system sends an inbound *S1SPLSHP-1.0* message.

Result: Maintenix updates the shipment record with the information from the vendor and changes the shipment status to IN-TRANS. Any quantity mismatches are handled when the shipment is received. If the received quantity is less than the requested quantity, Maintenix creates a new shipment record for the shortages.

Creating an order invoice in Maintenix

Goal: To create a new invoice for an order in Maintenix.

Summary: A new invoice is created in Maintenix.

Preconditions:

- The vendor is SPEC 2000 enabled. • The vendor has the S1INVCE command enabled. • The order must exist in Maintenix. • The order must have a status of ISSUED, PARTIAL, or RECEIVED. • The order line must exist in Maintenix.

Messaging: The external system sends an inbound *ATA_SparesInvoice-1.0* message. Maintenix returns an *ATA_InvoiceMessageAck-1.0* outbound message.

Result: A new invoice is created in Maintenix.

Updating an order invoice in Maintenix

Goal: To update an existing invoice for an order in Maintenix.

Summary: An invoice is updated in Maintenix.

Preconditions:

- The vendor is SPEC 2000 enabled.
- The vendor has the S1INVCE command enabled.
- The order must exist in Maintenix.
- The order must have a status of ISSUED, PARTIAL, or RECEIVED.
- The order line must exist in Maintenix.

Messaging: The external system sends an inbound *ATA_SparesInvoice-1.0* message. Maintenix returns an *ATA_InvoiceMessageAck-1.0* outbound message.

Result: The invoice is updated in Maintenix.

Updating a vendor's external key

Goal: To update the external key for a vendor in Maintenix.

Summary: If a new vendor is created in Maintenix, a external system can send a message to update the external key for that vendor.

Preconditions:

- The vendor exists in Maintenix.

Messaging: The external system sends an inbound *update_vendor_external_key-1.0* message.

Result: The external key is updated.

Adding part prices in Maintenix

Goal: To add new part price data for vendors Maintenix.

Summary: Part price data is typically uploaded in bulk from an external system into Maintenix. The bulk updates could range from weekly updates of special rates, monthly updates of seasonal rates, to yearly updates of vendor price catalogues. An external system sends a message to add new part price data for a vendors in Maintenix.

Preconditions:

- The vendor exists in Maintenix.
- The manufacturer exists in Maintenix.
- The part number exists in Maintenix.
- The unit of measure exists in Maintenix.
- The unit of measure applies to the part number in Maintenix.
- The currency exists in Maintenix.
- The vendor is a supplier of the part.

Messaging: The external system sends an inbound *MxPartPrice-1.0* message. Maintenix returns an *MxPartPriceResponse-1.0* outbound message.

Result: The part prices are added to Maintenix.

Updating part prices in Maintenix

Goal: To update part price data to Maintenix.

Summary: Part price data is typically uploaded in bulk from an external system into Maintenix. The bulk updates could range from weekly updates of special rates, monthly updates of seasonal rates, to yearly updates of vendor price catalogues. An external system sends a message to update part price data for a vendors in Maintenix.

Preconditions:

- The vendor exists in Maintenix.
- The manufacturer exists in Maintenix.
- The part number exists in Maintenix.
- The unit of measure exists in Maintenix.
- The unit of measure applies to the part number in Maintenix.
- The currency exists in Maintenix.
- The vendor is a supplier of the part.
- The part price exists in Maintenix.

Messaging: The external system sends an inbound *MxPartPrice-1.0* message. Maintenix returns an *MxPartPriceResponse-1.0* outbound message.

Result: The part prices in Maintenix are updated.

Marking part prices obsolete in Maintenix

Goal: To remove a part price from Maintenix.

Summary: Part price data is typically uploaded in bulk from an external system into Maintenix. The bulk updates could range from weekly updates of special rates, monthly updates of seasonal rates, to yearly updates of vendor price catalogues. An external system sends a message to mark existing part price data as obsolete.

Preconditions:

- The vendor exists in Maintenix.
- The manufacturer exists in Maintenix.
- The part number exists in Maintenix.
- The unit of measure exists in Maintenix.
- The unit of measure applies to the part number in Maintenix.
- The currency exists in Maintenix.
- The vendor is a supplier of the part.
- The part price exists in Maintenix.

Messaging: The external system sends an inbound *MxPartPrice-1.0* message. Maintenix returns an *MxPartPriceResponse-1.0* outbound message.

Result: The part prices in Maintenix are made obsolete.

Notification service scenarios

The Procurement API supports the following notification service scenarios.

Publishing vendors created in Maintenix

Goal: To publish the details of a new vendor to an external system automatically.

Summary: Maintenix sends a message to external systems when a new vendor is created in Maintenix.

Preconditions:

- A new vendor is created in Maintenix.

Messaging: Maintenix publishes a *vendor-created-1.1* outbound message. An external system can then send a *update_vendor_external_key-1.0* message to update the external key for the new vendor.

Result: Maintenix publishes a message for each new vendor that is created in Maintenix.

Publishing updates to a vendor's approval

Goal: To publish the changes to a vendor's approval to an external system automatically.

Summary: Maintenix sends a message to external systems when a vendor's approval is updated in Maintenix.

Preconditions:

- The approval status of a vendor is updated in Maintenix.

Messaging: Maintenix publishes a *vendor_approval_updated-1.0* outbound message.

Result: Maintenix publishes a message with the changes to a vendor's status.

4

Operational service messages

This section provides detailed information for each of the API operational messages. For each inbound message, the following information is provided:

- **Sample Messages:** A description of each of the sample messages that complete the supported scenarios.
- **Message Contents:** Information to help you understand how to use the message.
- **Mandatory Elements:** A table providing information about which elements are mandatory or data-specific in each of the supported scenarios. The following keys are used:
- • **M** indicates that the element is mandatory.
  - **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
  - **U** indicates that the element, if provided, must be unique within Maintenix
- **Errors and Exceptions:** A list of errors and exceptions that can be encountered when sending messages to Maintenix.
- **Mapping Elements to the Maintenix GUI:** A table providing the corresponding field in Maintenix for each of the elements contained in the message.

For each outbound message, a brief explanation is provided. Response messages are generally reduced to a few key elements that confirm the details of the successful inbound message. For more specific information about each of the response messages, review the XML schemas.

ATA_SparesQuoteRequest

The `ATA_SparesQuoteRequest` outbound message is sent to the external system to transmit an RFQ.

To enable this notification service message, you must provide the following `NAMESPACE` and the `ROOT_NAME` in Maintenix's INT_DEST_LOOKUP table.

- NAMESPACE: http://xml.mxi.com/xsd/procurement/ATA_SparesQuoteRequest/1.0
- ROOT_NAME: ATA_SparesQuoteRequest

You must also set the ACTIVE_BOOL to 1 for the following trigger Maintenix's UTL_TRIGGER table.

TRIGGER_ID	TRIGGER_CD	TRIGGER_NAME
99984	MX_RFQ_SEND	Send Request For Quotes

Sample messages

The following sample messages provide examples of the supported scenarios:

- Publishing_an_RFQ.xml: This message corresponds to the *Sending a request for quote (RFQ) from Maintenix* scenario. In this message, an RFQ is published from Maintenix.

Message triggers

The ATA_SparesQuoteRequest message is published to external systems when a new RFQ is created in Maintenix. Each message can contain a maximum of 10 part numbers.

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the ATA_SparesQuoteRequest message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<QTR>	RFQ Details page, Details group box, RFQ Number field.
<RequestHeader> <CIC>	RFQ Details page, Details tab, Spec 2000 Customer Code field.
<RequestHeader> <SPL>	Vendor page, Details group box, Vendor Code field.
<RequestDetails> <PNR>	RFQ Details page, RFQ Lines tab, Line Description column.
<RequestDetails> <QTQ>	RFQ Details page, RFQ Lines tab, Quantity column.

Element	Corresponding entry in Maintenix GUI
<RequestDetails> <PRI>	RFQ Details page, RFQ Lines tab, Priority column.
<RequestDetails> <SSD>	RFQ Details page, Details tab, Respond By field.

ATA_SparesQuoteInterim

The `ATA_SparesQuoteInterim` inbound message is sent by external systems acknowledge an RFQ in Maintenix.

The `ATA_SparesQuoteInterim.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Sending_Interim_Quote.xml`: This message corresponds to the *Sending a request for quote (RFQ) from Maintenix* scenario. In this message, an interim quote is sent from the external system after receiving a RFQ from Maintenix.

Message contents

The <CCD> element is always set to S1QUOTES.

Set the No Quote <NQT> element to Y, to indicate that you are not providing a final quote.

Mandatory elements

This section maps the mandatory elements for the `ATA_SparesQuoteInterim` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios Sending an RFQ from Maintenix
<CCD>	M
<QTR>	M
<ACK>	
<InterimHeader> <CIC>	M
<InterimHeader> <SPL>	M
<InterimHeader> <MessageSender>	
<InterimHeader> <MessageReceiver>	
<InterimDetails> <PNR>	M
<InterimDetails> <MFR>	
<InterimDetails> <NQT>	
<InterimDetails> <REM>	

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `ATA_SparesQuoteInterim` message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<QTR>	RFQ Details page, Details group box, RFQ Number field.
<ACK>	N/A.

Element	Corresponding entry in Maintenix GUI
<InterimHeader> <CIC>	RFQ Details page, Details tab, Spec 2000 Customer Code field.
<InterimHeader> <SPL>	RFQ Details page, Quote Comparison tab, Vendor column.
<InterimHeader> <MessageSender>	N/A.
<InterimHeader> <MessageReceiver>	N/A.
<InterimDetails> <PNR>	RFQ Details page, Quote Comparison tab, Line Description column.
<InterimDetails> <MFR>	N/A.
<InterimDetails> <NQT>	N/A.
<InterimDetails> <REM>	N/A.

ATA_SparesQuoteFinal

The `ATA_SparesQuoteFinal` inbound message is sent by external systems to respond to an RFQ in Maintenix.

The `ATA_SparesQuoteFinal.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Sending_Final_Quote.xml`: This message corresponds to the *Sending a request for quote (RFQ) from Maintenix* scenario. In this message, a final RFQ response is sent from the external system to Maintenix.

Message contents

The <CCD> element must be set to S1QUOTES to allow the vendor to be able to do quotation requests. You can enable S1QUOTES under the Communication tab on the Vendor Details page.

While the <IPN> element identifies the part number of the original RFQ, the <RPN> element identifies the replacing part number supplied by the vendor. If the vendor is not offering a replacement part, the mandatory <RPN> element must specify the same part number as the <IPN> element. If the vendor is offering a replacement part, and the <RPN> is different from the

<IPN> element, then the <RFC> element must specify the reason for the part number change, and the <INC> element must specify the interchangeability code.

If the replacement part <RPN> offered by a vendor exists in Maintenix, but is not an alternate of the IPN, Maintenix creates an unapproved alternate to the original part number, and alerts the engineering department. If the replacement part <RPN> offered by a vendor does not exist in Maintenix, then Maintenix creates an unapproved alternate to the original. Maintenix uses the <MFR>, if provided, as the manufacturer, and alerts the engineering department.

Mandatory elements

This section maps the mandatory elements for the `ATA_SparesQuoteFinal` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios
	Sending an RFQ from Maintenix
<CCD>	M
<QTR>	M
<ACK>	
<FinalHeader> <CIC>	M
<FinalHeader> <SPL>	M
<FinalHeader> <MessageSender>	
<FinalHeader> <MessageReceiver>	
<FinalDetails> <IPN>	M
<FinalDetails> <RPN>	M
<FinalDetails> <QTQ>	M
<FinalDetails> <KWD>	M
<FinalDetails> <UNP>	M

Element	Scenarios Sending an RFQ from Maintenix
<FinalDetails> <PFT>	M
<FinalDetails> <LTM>	M
<FinalDetails> <QTN>	M
<FinalDetails> <QTD>	M
<FinalDetails> <MSQ>	M

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `ATA_SparesQuoteFinal` message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<QTR>	RFQ Details page, Details group box, RFQ Number field.
<ACK>	N/A.
<FinalHeader> <CIC>	RFQ Details page, Details tab, Spec 2000 Customer Code field.
<FinalHeader> <SPL>	Vendor page, Details group box, Vendor Code field.
<FinalHeader> <MessageSender>	N/A.
<FinalHeader> <MessageReceiver>	N/A.
<FinalDetails> <IPN>	RFQ Details page, Quote Comparison tab, Line Description column.

Element	Corresponding entry in Maintenix GUI
<FinalDetails> <RPN>	RFQ Details page, Quote Comparison tab, Quoted Part column.
<FinalDetails> <QTQ>	RFQ Details page, Quote Comparison tab, Quote Qty column.
<FinalDetails> <KWD>	N/A.
<FinalDetails> <UNP>	RFQ Details page, Quote Comparison tab, Unit Price column.
<FinalDetails> <PFT>	RFQ Details page, Quote Comparison tab, Effective To column.
<FinalDetails> <LTM>	RFQ Details page, Quote Comparison tab, Lead Time column.
<FinalDetails> <QTN>	RFQ Details page, Quote Comparison tab, Quote Number column.
<FinalDetails> <QTD>	RFQ Details page, Quote Comparison tab, Quote Date column.

getIssuedPurchaseOrders-1.0

The `getIssuedPurchaseOrders` inbound message is sent by the external system to request orders that have been issued. Maintenix returns the requested information in an `S1CPOXMT` outbound message.

The `getIssuedPurchaseOrders-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Issued_Orders.xml`: This message corresponds to the *Requesting issued orders from Maintenix* scenario. In this message, a vendor code is sent to Maintenix.

Message contents

The external system sends a message specifying the vendor code of one vendor in Maintenix.

Mandatory elements

This section maps the mandatory elements for the `getIssuedPurchaseOrders` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios
	Requesting issues orders from Maintenix
<SUPPLIER>	M D

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `getIssuedPurchaseOrders` message elements.

Element	Corresponding entry in Maintenix GUI
<SUPPLIER>	Vendor Details page, Details group box, Vendor Code field.

S1CPOXMT-1.0

Maintenix sends an `S1CPOXMT` outbound message following an `getIssuedPurchaseOrders` inbound message from the external system.

The `S1CPOXMT-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Issued_Orders_Response.xml`: This message corresponds to the *Requesting issued orders from Maintenix* scenario. In this message, Maintenix returns the list of issued orders that were requested in the `Requesting_Issued_Orders.xml` message.

Message contents

In the `S1CPOXMT` message, the TNC code uses the following format.

The first character in the TNC code has one of the following values:

Code	Description
3	Normal order
L	Normal confirming order
4	Expedite order
M	Expedite confirming order

The second character in the TNC code may have the following values:

Code	Description
2	Initial provisioning item
3	New stock item
4	Replenishment stock item
5	Allocation item
6	Special requirement item
9	Special business agreement

The third character in the TNC code may have the following values:

Code	Description
1	Routine Action
2	Acceptance of a Quotation
3	Alteration to an order

Code	Description
6	Cancel or decrease an order
8	Increase an order
9	Request partial ship quantity of an open order

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `S1CPOXMT` message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<CIC>	N/A.
<SPL>	Order Details page, Order Information group box, Vendor field.
<ICR>	Order Details page, Details group box. The currency code of the Total Price field.
<POC>	Order Details page, Order Lines tab. The total number of order lines.
<S1CPOXMT_ORDER> <TNC>	N/A.
<S1CPOXMT_ORDER> <CPO>	Order Details page, Order Number field combined with Order Details page, Order Lines tab, Line No column.
<S1CPOXMT_ORDER> <PNR>	Part Details page, OEM Part Number field.
<S1CPOXMT_ORDER> <QTO>	Order Details page, Order Lines tab, QTY column.
<S1CPOXMT_ORDER> <UNT>	Order Details page, Order Lines tab, Unit Price column.
<S1CPOXMT_ORDER> <UNP>	Order Details page, Order Lines tab, Unit Price column.
<S1CPOXMT_ORDER> <SSD>	Order Details page, Order Lines tab, Date column.

Element	Corresponding entry in Maintenix GUI
<S1CPOXMT_ORDER> <SHT>	Order Details page, Details tab, Ship To field.
<S1CPOXMT_ORDER> <SDC>	N/A.
<S1CPOXMT_ORDER> <MFR>	Part Details page, Manufacturer field.
<S1CPOXMT_ORDER> <PRI>	Order Details page, Details tab, Priority field.
<S1CPOXMT_ORDER> <ACN>	
<S1CPOXMT_ORDER> <CTN>	

S1CPOACK-1.0

The `S1CPOACK` inbound message is sent by external systems to acknowledge an order in Maintenix.

The `S1CPOACK-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Acknowledging_Order.xml`: This message corresponds to the *Requesting issued orders from Maintenix* scenario. In this message, an external system acknowledges the receipt of an order form Maintenix.

Message contents

The external system sends a message specifying the details of the orders that was sent from Maintenix. Maintenix will transition the status of the order to acknowledged.

Mandatory elements

This section maps the mandatory elements for the `S1CPOACK` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenario
	Sending shipment advisory
<CCD>	M D
<CIC>	M D
<SPL>	M D
<POC>	M
<LPO>	M D

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `S1CP0ACK` message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<CIC>	N/A.
<SPL>	Order Details page, Order Information group box, Vendor field.
<POC>	Order Details page, Order Lines tab. The total number of order lines.
<LPO>	Order Details page, Order Number field combined with Order Details page, Order Lines tab, Line No column.

S1EXCXMT-1.0

The S1EXCXMT inbound message is sent by external systems to send exceptions to an order in Maintenix.

The S1EXCXMT-1.0.xsd schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- Sending_Order_Exceptions.xml: This message corresponds to the *Sending order exceptions to Maintenix* scenario. In this message, an order is identified, the quantity is increased by 3 units, and the shipping date as changed.

Message contents

In the S1EXCXMT message, you must identify the order the you wish to change. You can then update any of the following details:

- Either the increase <ICQ)> or the decrease <DCQ> in the order quantity.
- The shipping date.
- The part number of the item.

Maintenix records the details of the change request on the History tab of the Order Details page. Maintenix does not automatically change the order directly. You can assign a **Purchase Order Exception** alert to the purchasing agent role to notify users when Maintenix receives a request from an external system to modify an order. If necessary, the purchasing agent can then modify the details of the order manually.

Mandatory elements

This section maps the mandatory elements for the S1EXCXMT message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenario Sending order exceptions to Maintenix
<CCD>	M D
<CIC>	M D
<SPL>	M D
<POC>	M
<S1EXCXMT_ORDER> <CPO>	M D
<S1EXCXMT_ORDER> <PNR>	M D
<S1EXCXMT_ORDER> <ICQ>	
<S1EXCXMT_ORDER> <DCQ>	
<S1EXCXMT_ORDER> <QTO>	
<S1EXCXMT_ORDER> <UNT>	
<S1EXCXMT_ORDER> <SCD>	
<S1EXCXMT_ORDER> <REM>	
<S1EXCXMT_ORDER> <S1EXCXMT_RPN_INC> <RPN>	
<S1EXCXMT_ORDER> <S1EXCXMT_RPN_INC> <INC>	
<S1EXCXMT_ORDER> <S1EXCXMT_RPN_INC> <MFR>	
<S1EXCXMT_ORDER> <S1EXCXMT_UNP_ICR> <UNP>	
<S1EXCXMT_ORDER> <S1EXCXMT_UNP_ICR> <ICR>	

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.

- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the S1EXCXMT message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<CIC>	N/A.
<SPL>	Order Details page, Order Information group box, Vendor field.
<POC>	Order Details page, Order Lines tab. The total number of order lines.
<S1EXCXMT_ORDER> <CPO>	Order Details page, Order Number field combined with Order Details page, Order Lines tab, Line No column.
<S1EXCXMT_ORDER> <PNR>	Part Details page, Part group box, OEM Part Number field.
<S1EXCXMT_ORDER> <ICQ>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <QTO>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <DCQ>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <UNT>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <SCD>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <REM>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <S1EXCXMT_RPN_INC> <RPN>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <S1EXCXMT_RPN_INC> <INC>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <S1EXCXMT_RPN_INC> <MFR>	Order Details page, History tab, Notes column.

Element	Corresponding entry in Maintenix GUI
<S1EXCXMT_ORDER> <S1EXCXMT_UNP_ICR> <UNP>	Order Details page, History tab, Notes column.
<S1EXCXMT_ORDER> <S1EXCXMT_UNP_ICR> <ICR>	Order Details page, History tab, Notes column.

S1SPLSHP-1.0

The S1SPLSHP inbound message is sent by external systems send a shipment advisory to Maintenix.

The S1SPLSHP-1.0.xsd schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- Requesting_Changes_to_Order.xml: This message corresponds to the *Sending shipment advisory to Maintenix* scenario. In this message, a shipment advisory is sent.

Message contents

The S1SPLSHP message identifies the shipment in Maintenix using the <CPO> element, which specifies the originating order number and line number, and the <PNR> element, which specifies the OEM part number of the item. If no corresponding order is found in Maintenix, and error is thrown.

Additional information about the shipment including the quantity, shipment date, waybill number, carrier name, and the size and weight details, are used to update the shipment record in Maintenix.

When Maintenix receives a shipment advisory message, the status of the shipment is updated to intransit (IN-TRANS). If the shipment status in Maintenix is already in-transit, the shipment advisory message is ignored by Maintenix.

You can assign a **Shipment Advisory** alert to notify roles when Maintenix receives a shipment advisory message from an external system, or a **No Matching Shipment Record for Shipment Advisory** alert to notify roles when Maintenix receives a shipment advisory message from an external system that does not match an existing shipment in Maintenix.

Mandatory elements

This section maps the mandatory elements for the S1SPLSHP message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenario
	Sending shipment advisory
<CCD>	M
<CIC>	M
<SPL>	M D
<CPO>	M D
<PNR>	M D
<SHQ>	M
<UNT>	M D
<SHD>	M
<BOL>	
<BLS>	
<BLW>	
<PRI>	
<CAR>	
<BOX>	
<SWT>	
<SHM>	D
<REM>	

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the S1SPLSHP message elements.

Element	Corresponding entry in Maintenix GUI
<CCD>	N/A.
<CIC>	N/A.
<SPL>	Order Details page, Order Information group box, Vendor field.
<CPO>	Order Details page, Order Information group box, Order Number field, and the order line number.
<PNR>	Part Details page, Part group box, OEM Part Number field.
<SHQ>	Shipment Details page, Shipment Lines tab, Expected Qty column.
<UNT>	Shipment Details page, Shipment Lines tab, Expected Qty column.
<SHD>	Shipment Details page, Details tab, Shipment Date field.
<BOL>	Shipment Details page, Details tab, Waybill Number field.
<BLS>	Shipment Details page, History tab, Note column.
<BLW>	Shipment Details page, Details tab, Weight field.

Element	Corresponding entry in Maintenix GUI
<PRI>	Shipment Details page, Identification group box, Priority field.
<CAR>	Shipment Details page, Details tab, Carrier Name field.
<BOX>	Shipment Details page, History tab, Note column.
<SWT>	Shipment Details page, History tab, Note column.
<SHM>	Shipment Details page, Details tab, Transportation Type field.
<REM>	Shipment Details page, History tab, Note column.

ATA_SparesInvoice-1.0

The `ATA_SparesInvoice` inbound message is sent by external systems to create or edit order invoices in Maintenix.

The `ATA_SparesInvoice-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Creating_an_Invoice.xml`: This message corresponds to the *Creating an order invoice in Maintenix* scenario. In this message, a new invoice record is created in Maintenix.
- `Updating_an_Invoice.xml`: This message corresponds to the *Updating an order invoice in Maintenix* scenario. In this message, the invoice record created in the previous sample is updated.

Message contents

When updating an invoice the <ADS> Adjustment Sequence Number must be incremented higher than the previous message that we sent for this invoice or the API will not update the invoice. No exception will be thrown indicating that the data in the message was ignored.

Mandatory elements

This section maps the mandatory elements for the `ATA_SparesInvoice` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios	
	Creating an order invoice	Updating an order invoice
<InvoiceHeader> <CIC>	M D	M D
<InvoiceHeader> <SPL>	M D	M D
<InvoiceHeader> <MessageSender>	M	M
<InvoiceHeader> <MessgeReceiver>	M	M
<InvoiceDetails> <CPO>	M D	M D
<InvoiceDetails> <PNR>	M D	M D
<InvoiceDetails> <INV>	M	M
<InvoiceDetails> <INQ>	M D	M D
<InvoiceDetails> <UPN>	M D	M D
<InvoiceDetails> <IND>	M	M
<InvoiceDetails> <INA>	M	M
<InvoiceDetails> <ITC>	M	M
<InvoiceDetails> <PAY>	M	M
<InvoiceDetails> <SHT>	M D	M D
<InvoiceDetails> <PSN>	M	M
<InvoiceDetails> <BOL>		
<InvoiceDetails> <CDT>		

Element	Scenarios	
	Creating an order invoice	Updating an order invoice
<InvoiceDetails> <OCH>		
<InvoiceDetails> <PDP>		
<InvoiceDetails> <TaxAmountData>		
<InvoiceDetails> <TRN>		
<InvoiceDetails> <SPD>		
<InvoiceDetails> <TAX>		
<InvoiceDetails> <ADS>		
<InvoiceDetails> <RIN>		
<InvoiceDetails> <HTC>		
<InvoiceDetails> <VRN>		
<InvoiceDetails> <TXD>		
<InvoiceDetails> <REM>		

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `ATA_SparesInvoice` message elements.

Element	Corresponding entry in Maintenix GUI
<InvoiceHeader> <CIC>	PO Details page, Order Details tab, Order Details area, Customer Code field.

Element	Corresponding entry in Maintenix GUI
<InvoiceHeader> <SPL>	PO Details page, Order Information Area, Vendor Code field.
<InvoiceHeader> <MessageSender>	N/A
<InvoiceHeader> <MessgeReceiver>	N/A
<InvoiceDetails> <CPO>	PO Details page, Order Information area, Order Number field. PO Details page, Order Information area, Line Number field. **Note** This field consists of the order number and line number, concatenated with a hyphen in the following format: [Order Number]-[Line Number].
<InvoiceDetails> <PNR>	PO Invoice Lines tab, Line Description column.
<InvoiceDetails> <INV>	PO Details page, PO Invoice Information area, PO Invoice Number field.
<InvoiceDetails> <INQ>	PO Details page, PO Invoice Lines tab, Qty column.
<InvoiceDetails> <UPN>	PO Details page, PO Invoice Lines tab, Unit Price column.
<InvoiceDetails> <IND>	PO Invoice Details page, Details tab, Details area, PO Invoice Date field.
<InvoiceDetails> <INA>	N/A
<InvoiceDetails> <ITC>	N/A
<InvoiceDetails> <PAY>	N/A
<InvoiceDetails> <SHT>	PO Details page, Details tab, Shipping and Terms area, Ship To code.
<InvoiceDetails> <PSN>	PO Invoice Details page, Details tab, Shipment and Terms area, Packing Sheet Number field.

Element	Corresponding entry in Maintenix GUI
<InvoiceDetails> <BOL>	PO Invoice Details page, Details tab, Shipment and Terms area, Bill of Lading Number field.
<InvoiceDetails> <CDT><CDC>	PO Invoice Details page, Details tab, Cash Discount area, Cash Discount field. **Note** Maintenix displays the highest discount percentage from this element.
<InvoiceDetails><CDT><CDM>	PO Invoice Details page, Details tab, Cash Discount area, Discount Expiry Date field. **Note** Maintenix displays this information if CDT.CDC is provided. This integer value is added to the vendor invoice date (IND) to populate the expiry date.
<InvoiceDetails> <OCH>	PO Invoice Details page, Details tab, Taxes and Other ChParges area, Other Charges table.
<InvoiceDetails> <PDP>	N/A.
<InvoiceDetails> <TaxAmountData>	PO Invoice Details page, PO Invoice Lines tab, Unit Price column.
<InvoiceDetails> <TRN>	N/A.
<InvoiceDetails> <SPD>	PO Invoice Details page, PO Invoice Lines tab, Tax Point Date column.
<InvoiceDetails> <TAX>	N/A.
<InvoiceDetails> <ADS>	PO Invoice Details page, Details tab, Miscellaneous area, Adjustment Sequence Number field.
<InvoiceDetails> <RIN>	PO Invoice Details page, Details tab, Miscellaneous area, Reference Invoice Number field.

Element	Corresponding entry in Maintenix GUI
<InvoiceDetails> <HTC>	PO Invoice Details page, Details tab, Miscellaneous area, Harmonized Tariff Code field.
<InvoiceDetails> <VRN>	N/A.
<InvoiceDetails> <TXD>	N/A.
<InvoiceDetails> <REM>	PO Invoice Details, PO Invoice Lines tab, Note column.

ATA_InvoiceMessageAck-1.0

The `ATA_InvoiceMessageAck` outbound message is returned to the external system upon a successful ATA_SparesInvoice inbound message.

The `ATA_InvoiceMessageAck-1.0.xsd` schema contains annotations describing each element.

MxPartPrice-1.0

The `MxPartPrice` inbound message is sent by external systems to bulk create or edit order part prices in Maintenix.

The `MxPartPrice-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Adding_Part_Prices.xml`: This message corresponds to the *Adding part prices in Maintenix* scenario. In this message, a part price is added to Maintenix.
- `Updating_Part_Prices.xml`: This message corresponds to the *Updating part prices in Maintenix* scenario. In this message, the part price added in the previous sample message is updated.
- `Marking_Part_Prices_Obsolete.xml`: This message corresponds to the *Marking part prices obsolete in Maintenix* scenario. In this message, the part price added in the previous sample message is made obsolete.

Message contents

The possible values for the <CHG> element are as follows:

- N: The part price is new.
- R: The part price is revised.
- D: The part price is deleted.

The N and R commands are interchangeable: regardless of which command is specified in a message, a part price is created if it does not exist in Maintenix, and a part price is updated if it does exist in Maintenix.

For each vendor, a part can have any number of prices and effective dates but no exact duplicates. Therefore, the following elements determine whether the part price specified in the message exists in Maintenix:

- <SPL> The vendor code.
- <MFR> The manufacturer code.
- <PNR> The part number.
- <PED> The effective from date.
- <MSQ> The minimum order quantity.

If the values for these elements matches an existing vendor part price exactly, then the existing part price is updated with the other values in the message. If the values for these elements do not match an existing vendor part price, then a new part price is created in Maintenix.

There can be only one active Contract Price.

Mandatory elements

This section maps the mandatory elements for the `MxPartPrice` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios		
	Adding part prices	Updating part prices	Marking part prices obsolete
<SPL>	M D	M D	M D
<CHG>	M	M	M

Element	Scenarios		
	Adding part prices	Updating part prices	Marking part prices obsolete
<MFR>	M D	M D	M D
<PNR>	M D	M D	M D
<UNP>	M	M	M
<UNT>	M D	M D	M D
<ICR>	M D	M D	M D
<PED>	M	M D	M D
<LMT>	M	M	M
<PFD>			
<PDP>			
<MSQ>	M	M D	M D
<SPQ>			
<PTP>			
<CNR>			
<QRF>			
<RFQ>	D	D	D
<VNT>			

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `MxPartPrice` message elements.

Element	Corresponding entry in Maintenix GUI
<SPL>	Vendor Part Price Details page, Part Price group box, Vendor field.
<CHG>	N/A
<MFR>	Vendor Part Price Details page, Part Price group box, Manufacturer field.
<PNR>	Vendor Part Price Details page, Part Price group box, OEM Part Number field.
<UNP>	Vendor Part Price Details page, Unit Price column.
<UNT>	Vendor Part Price Details page, Unit Price column.
<ICR>	Vendor Part Price Details page, Unit Price column.
<PED>	Vendor Part Price Details page, Effective From column.
<LMT>	Vendor Part Price Details page, Lead Time column.
<PFD>	Vendor Part Price Details page, Effective To column.
<PDP>	Vendor Part Price Details page, Discount Unit Price column.
<MSQ>	Vendor Part Price Details page, Min column.
<SPQ>	Vendor Part Price Details page, Standard column.
<PTP>	Vendor Part Price Details page, Price Type column.
<CNR>	Vendor Part Price Details page, Contract Number column.
<QRF>	Vendor Part Price Details page, Quote Reference column.

Element	Corresponding entry in Maintenix GUI
<RFQ>	Vendor Part Price Details page, RFQ column.
<VNT>	Vendor Part Price Details page, Vendor Note column.

MxPartPriceResponse-1.0

The `MxPartPriceResponse` outbound message is returned to the external system upon a successful `MxPartPrice` inbound message.

The `MxPartPriceResponse-1.0.xsd` schema contains annotations describing each element.

update_vendor_external_key-1.0

The `update_vendor_external_key` inbound message is sent by external systems to update the external key for a vendor in Maintenix.

The `update_vendor_external_key-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Updating_Vendor_External_Key.xml`: This message corresponds to the *Updating a vendor's external key* scenario. In this message, the external key for a vendor is updated.

Message contents

If the vendor has an external key already, then this message will update the existing external key. If the vendor does not have an external key, then this message will create the external key for the vendor.

Mandatory elements

This section maps the mandatory elements for the `update_vendor_external_key` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.

- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios
	Update a vendor's external key
<code>	M D
<external_key>	M

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `create-vendor-request` message elements.

Element	Corresponding entry in Maintenix GUI
<code>	Vendor Details page, Details group box, Vendor Code field.
<external_key>	Vendor Details page, Details tab, External Key field.

create-vendor-request-1.0

The `create-vendor-request` inbound message is sent by external systems to create vendors in Maintenix.

The `create-vendor-request-1.0.xsd` schema contains annotations describing each element.

Endpoint: http://<maintenix_server>:<http_port>/procurement/VendorService

WSDL link: http://<maintenix_server>:<http_port>/procurement/VendorService?wsdl

Sample messages

The following sample messages provide examples of creating a vendor:

- `Creating_Vendor_Request.xml`

Message contents

This message will create a vendor if the provided vendor code does not exist in Maintenix.

> **Note** For this particular message, you do not need to include the request section in the SOAP message when sending it through HTTP.

Mandatory elements

This section maps the mandatory elements for the create-vendor-request message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios
	Create vendors
<code>	M U
<name>	M
<organization-code>	M
<certificate-number>	
<certificate-expiry-date>	
<type-code>	M D
<approval-type-code>	D
<terms-and-conditions>	D
<spec-2000-enabled>	
<currency-code>	M D

Element	Scenarios
	Create vendors
<min-purchase-amount>	
<external-key>	
<standard-borrow-rate>	D
<default-airport>	D
<time-zone>	M D
<shipping-address> <address1>	M
<shipping-address> <address2>	
<shipping-address> <city>	M
<shipping-address> <country>	M
<shipping-address> <state>	
<shipping-address> <zip>	M
<main-contact-info> <name>	M
<main-contact-info> <job-title>	
<main-contact-info> <phone-number>	
<main-contact-info> <fax-number>	
<main-contact-info> <email-address>	
<notes>	
<note-to-receiver>	

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `create-vendor-request` message elements.

Element	Corresponding entry in Maintenix GUI
<code>	Vendor Details page, Details group box, Vendor Code field.
<name>	Vendor Details page, Details group box, Vendor Name field.
<organization-code>	N/A
<certificate-number>	Vendor Details page, Details tab, Details group box, Certificate Number field.
<certificate-expiry-date>	Vendor Details page, Details tab, Details group box, Certificate Expiry field.
<type-code>	Vendor Details page, Details tab, Details group box, Type field.
<approval-type-code>	Vendor Details page, Details tab, Details group box, Approval Type field.
<terms-and-conditions>	Vendor Details page, Details tab, Details group box, Terms & Conditions field.
<spec-2000-enabled>	Vendor Details page, Details tab, Details group box, Spec 2000 Enabled check box.
<currency-code>	Vendor Details page, Details tab, Details group box, Currency field.
<min-purchase-amount>	Vendor Details page, Details tab, Details group box, Min Purchase Amount field.
<external-key>	Vendor Details page, Details tab, Details group box, External Key field.
<standard-borrow-rate>	Vendor Details page, Details tab, Details group box, Standard Borrow Rate field.
<default-airport>	Vendor Details page, Details tab, Airports group box, Airport / Default column.

Element	Corresponding entry in Maintenix GUI
<time-zone>	Vendor Details page, Details tab, Details group box, Time Zone field.
<shipping-address> <address1>	Vendor Details page, Details tab, Shipping Address group box, Address field.
<shipping-address> <address2>	Vendor Details page, Details tab, Shipping Address group box, Address field.
<shipping-address> <city>	Vendor Details page, Details tab, Shipping Address group box, City field.
<shipping-address> <country>	Vendor Details page, Details tab, Shipping Address group box, Country field.
<shipping-address> <state>	Vendor Details page, Details tab, Shipping Address group box, State field.
<shipping-address> <zip>	Vendor Details page, Details tab, Shipping Address group box, ZIP/Postal Code field.
<main-contact-info> <name>	Vendor Details page, Communication tab, Contacts group box, Name column.
<main-contact-info> <job-title>	Vendor Details page, Communication tab, Contacts group box, Job Title column.
<main-contact-info> <phone-number>	Vendor Details page, Communication tab, Contacts group box, Phone column.
<main-contact-info> <fax-number>	Vendor Details page, Communication tab, Contacts group box, Fax column.
<main-contact-info> <email-address>	Vendor Details page, Communication tab, Contacts group box, Email column.
<notes>	Vendor Details page, Details tab, Vendor Notes group box.
<note-to-receiver>	Vendor Details page, Details tab, Note to Receiver group box.

create-vendor-response-1.0

The `create-vendor-response` outbound message is returned to the external system upon a successful `create-vendor-request` inbound message.

The `create-vendor-response-1.0.xsd` schema contains annotations describing each element.

update-vendor-request-1.0

The `update-vendor-request` inbound message is sent by external systems to update vendors in Maintenix.

The `update-vendor-request-1.0.xsd` schema contains annotations describing each element.

Endpoint: http://<maintenix_server>:<http_port>/procurement/VendorService

WSDL link: http://<maintenix_server>:<http_port>/procurement/VendorService?wsdl

Message contents

This message will update a vendor if the provided vendor code or external key exists in Maintenix.

> **Note** For this particular message, you do not need to include the request section in the SOAP message when sending it through HTTP.

Mandatory elements

This section maps the mandatory elements for the `update-vendor-request` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios
	Create vendors
<code>	M D U

Element	Scenarios
	Create vendors
<external key>	M D U
<name>	
<organization-code>	D
<certificate-number>	
<certificate-expiry-date>	
<type-code>	D
<approval-type-code>	D
<terms-and-conditions>	D
<spec-2000-enabled>	
<currency-code>	D
<min-purchase-amount>	
<standard-borrow-rate>	D
<time-zone>	D
<shipping-address> <address1>	
<shipping-address> <address2>	
<shipping-address> <city>	
<shipping-address> <country>	D
<shipping-address> <state>	D
<shipping-address> <zip>	
<main-contact-info> <name>	
<main-contact-info> <job-title>	
<main-contact-info> <phone-number>	
<main-contact-info> <fax-number>	
<main-contact-info> <email-address>	

Element	Scenarios Create vendors
<notes>	
<note-to-receiver>	

Errors and exceptions

The following errors and exceptions can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `update-vendor-request` message elements.

Element	Corresponding entry in Maintenix GUI
<code>	Vendor Details page, Details group box, Vendor Code field.
<name>	Vendor Details page, Details group box, Vendor Name field.
<organization-code>	N/A
<certificate-number>	Vendor Details page, Details tab, Details group box, Certificate Number field.
<certificate-expiry-date>	Vendor Details page, Details tab, Details group box, Certificate Expiry field.
<type-code>	Vendor Details page, Details tab, Details group box, Type field.
<approval-type-code>	Vendor Details page, Details tab, Details group box, Approval Type field.
<terms-and-conditions>	Vendor Details page, Details tab, Details group box, Terms & Conditions field.

Element	Corresponding entry in Maintenix GUI
<spec-2000-enabled>	Vendor Details page, Details tab, Details group box, Spec 2000 Enabled check box.
<currency-code>	Vendor Details page, Details tab, Details group box, Currency field.
<min-purchase-amount>	Vendor Details page, Details tab, Details group box, Min Purchase Amount field.
<external-key>	Vendor Details page, Details tab, Details group box, External Key field.
<standard-borrow-rate>	Vendor Details page, Details tab, Details group box, Standard Borrow Rate field.
<default-airport>	Vendor Details page, Details tab, Airports group box, Airport / Default column.
<time-zone>	Vendor Details page, Details tab, Details group box, Time Zone field.
<shipping-address> <address1>	Vendor Details page, Details tab, Shipping Address group box, Address field.
<shipping-address> <address2>	Vendor Details page, Details tab, Shipping Address group box, Address field.
<shipping-address> <city>	Vendor Details page, Details tab, Shipping Address group box, City field.
<shipping-address> <country>	Vendor Details page, Details tab, Shipping Address group box, Country field.
<shipping-address> <state>	Vendor Details page, Details tab, Shipping Address group box, State field.
<shipping-address> <zip>	Vendor Details page, Details tab, Shipping Address group box, ZIP/Postal Code field.
<main-contact-info> <name>	Vendor Details page, Communication tab, Contacts group box, Name column.
<main-contact-info> <job-title>	Vendor Details page, Communication tab, Contacts group box, Job Title column.

Element	Corresponding entry in Maintenix GUI
<main-contact-info> <phone-number>	Vendor Details page, Communication tab, Contacts group box, Phone column.
<main-contact-info> <fax-number>	Vendor Details page, Communication tab, Contacts group box, Fax column.
<main-contact-info> <email-address>	Vendor Details page, Communication tab, Contacts group box, Email column.
<notes>	Vendor Details page, Details tab, Vendor Notes group box.
<note-to-receiver>	Vendor Details page, Details tab, Note to Receiver group box.

update-vendor-response-1.0

The `update-vendor-response` outbound message is returned to the external system upon a successful update-vendor-request inbound message.

The `update-vendor-response-1.0.xsd` schema contains annotations describing each element.

unapprove-vendor-request-1.0

The `unapprove-vendor-request` inbound message is sent by external systems to unapprove vendors in Maintenix.

The `unapprove-vendor-request-1.0.xsd` schema contains annotations describing each element.

Endpoint: http://<maintenix_server>:<http_port>/procurement/VendorService

WSDL link: http://<maintenix_server>:<http_port>/procurement/VendorService?wsdl

Message contents

This message will unapprove a vendor if the provided vendor code / external key and organization code exist in Maintenix.

Note For this particular message, you do not need to include the request section in the SOAP message when sending it through HTTP.

Mandatory elements

This section maps the mandatory elements for the `unapprove-vendor-request` message in each supported scenario.

The following keys are used:

- **M** indicates that the element is mandatory.
- **D** indicates that the element, if provided, depends on the existence of corresponding data or reference terms in Maintenix.
- **U** indicates that the element, if provided, must be unique within Maintenix.

Element	Scenarios
	unapprove vendors
<code>	M D U
<external-key>	M D U
<organization-code>	M D
<reason-code>	D
<note>	

Errors and exceptions

The following is a list of errors and exceptions that can occur when processing a message:

- Invalid XML message formatting.
- Elements marked as mandatory are not provided.
- Elements that must correspond to existing data or reference terms in Maintenix are invalid.
- Vendor code or external key does not exist in Maintenix.
- More than one vendors have the same external key.

Mapping elements to Maintenix

The following table displays the corresponding field in the Maintenix user interface for each of the `unapprove-vendor-request` message elements.

Element	Corresponding entry in Maintenix GUI
<code>	Approval Details page, Sub title.
<external-key>	N/A
<organization-code>	Approval Details page, Sub title.
<reason-code>	Approval Details page, Order / Service Types table, Reason column.
<note>	Approval Details page, Order / Service Types table, Note column.

unapprove-vendor-response-1.0

The `unapprove-vendor-response` outbound message is returned to the external system upon a successful unapprove-vendor-request inbound message.

The `unapprove-vendor-response-1.0.xsd` schema contains annotations describing each element.

5

Notification service messages

The following chapter provides detailed information for each of the API notification messages.

For each message, the following information is provided:

- **Sample Messages:** A description of each of the sample messages that complete the supported scenarios.
- **Message Triggers:** A list of events that trigger the outbound notification message.
- **Mapping Elements to the Maintenix GUI:** A table with the corresponding fields in Maintenix for each element.

vendor-created-1.1

The `vendor-created` outbound message is sent to the external system.

To enable this notification service message, you must set the `ACTIVE_BOOL` to 1 for the following trigger Maintenix's UTL_TRIGGER table.

TRIGGER_ID	TRIGGER_CD	TRIGGER_NAME
99915	MX_VENDOR_CREATE	Vendor Creation

The `vendor-created-1.1.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Publishing_Vendor_Created.xml`: This message corresponds to the *Publishing vendors created in Maintenix* scenario. In this message, the external system is notified that a vendor has been created in Maintenix.

Message triggers

The `vendor-created` message is published to external systems when a new vendor is created in Maintenix.

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `vendor-created` message elements.

Element	Corresponding entry in Maintenix GUI
<code>	Vendor Details page, Details group box, Vendor Code field.
<name>	Vendor Details page, Details group box, Vendor Name field.
<organization>	Organization Details page, Identification group box, Code field.
<certificate_number>	Vendor Details page, Details tab, Certificate Number field.
<certificate_expiry>	Vendor Details page, Details tab, Certificate Expiry field.
<vendor_type>	Vendor Details page, Details tab, Type field.
<approval_type>	Vendor Details page, Details tab, Approval Type field.
<terms>	Vendor Details page, Details tab, Terms & Conditions field.
<spec2000_flag>	Vendor Details page, Details tab, Spec 2000 Enabled check box.
<currency>	Vendor Details page, Details tab, Currency field.
<min_order_amount>	Vendor Details page, Details tab, Min Purchase Amount field.
<external_key>	Vendor Details page, Details tab, External Key field.

Element	Corresponding entry in Maintenix GUI
<std_borrow_rate>	Vendor Details page, Details tab, Standard Borrow Rate field.
<default_airport>	Vendor Details page, Details tab, Airports group box.
<time_zone>	Vendor Details page, Details tab, Time Zone field.
<shipping_address>	Vendor Details page, Details tab, Shipping Address group box.
<contact_info>	Vendor Details page, Communication tab, Contacts group box.
<notes>	Vendor Details page, Details tab, Vendor Notes group box.
<notes_to_receiver>	Vendor Details page, Details tab, Note To Receiver group box.

vendor_approval_updated-1.0

The `vendor_approval_updated` outbound message is sent to the external system.

To enable this notification service message, you must set the `ACTIVE_BOOL` to 1 for the following trigger Maintenix's UTL_TRIGGER table.

TRIGGER_ID	TRIGGER_CD	TRIGGER_NAME
99955	MX_VENDOR_APPROVAL_UPDATE	Vendor Approval Update

The `vendor_approval_updated-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Publishing_Vendor_Approval_Updated.xml`: This message corresponds to the *Publishing updates to a vendor's approval* scenario. In this message, the external system is notified that a vendor's approval has been updated in Maintenix.

Message triggers

The `vendor_approval_updated` message is published to external systems when a vendor's approval is updated in Maintenix.

Mapping elements to Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `vendor_approval_updated` message elements.

Element	Corresponding entry in Maintenix GUI
<code>	Vendor Details page, Details group box, Vendor Code field.
<organization>	
<external_key>	Vendor Details page, Details tab, External Key field.
<approvals> <approval> <order_type>	Approval Details page, Order Type column.
<approvals> <approval> <approval_status>	Approval Details page, Status column.
<approvals> <approval> <approval_reason>	Approval Details page, Reason column.
<approvals> <approval> <approval_note>	Approval Details page, Note column.
<approvals> <approval> <warning_note>	Approval Details page, Warning Note column.
<approvals> <approval> <approval_expiry>	Approval Details page, Approval Expiry column.

6

Integration framework

External system developers who are developing software to integrate their system with Maintenix, can use Maintenix Adapters. You have the option to use the adapters to integrate directly with Maintenix or to include the IFS Maintenix Integration Framework in the integration framework.

Maintenix Adapters use a Web service to integrate with an external system. The clients and servers exchange XML (Extensible Markup Language) messages that follow the SOAP (Service Oriented Architecture Protocol) version 1.1 standard.

The external system can access the Web services using SoapScope or another Web service tool.

You can find a copy of the WSDL (Web Services Description Language) request definition (Request.wsdl) in the Maintenix download directory, in the am-docs\Integrators and Developers\ Integration Resources folder, or at the following URL: `http://<username>:<password>@<servername>:7001/mxcorews/services/RequestWS?wsdl`

External systems can interact with Maintenix using the following methods:

- Operational Service Messages are used to send data to Maintenix. Maintenix processes the data, and then returns information about the success or failure of the message.
- Querying Service Messages are used to retrieve information from Maintenix.
- Notification Service Messages are generated by Maintenix, and sent to a Web service on the external system. These messages notify the external system when important events occur within Maintenix.

Operational and querying service messages

Use operational service messages when your external system must send information to Maintenix for processing. An example of this type of messaging is when an external system must tell Maintenix that a flight occurred or that a work package was completed.

Use the querying service messages when your external system must query or extract information from Maintenix. An example of this type of messaging is when an external system needs to know the status of an aircraft in Maintenix, or when a finance system needs to know the list of financial transactions that have occurred in Maintenix.

All operational and querying service messages are invoked and handled in the same way. The external system calls the following generic HTTP Web service: `http://<username>:<password>@<servername>:<port>/mxcorews/services/RequestWS`

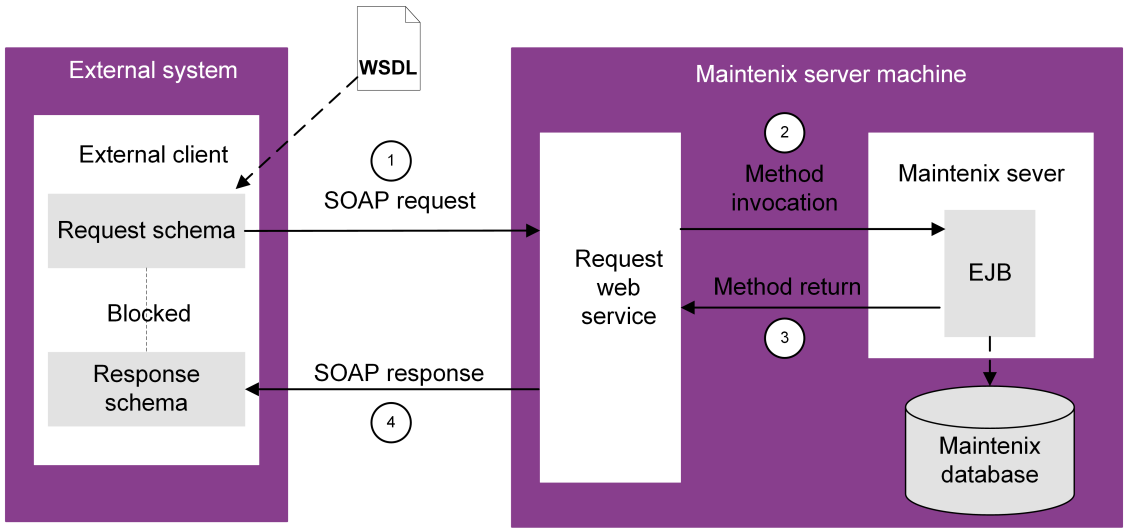
The same Web service is called, regardless of the data that is passed.

Maintenix then looks at the content of the HTTP request's <soap:Body> element to determine the logic that it should execute. Maintenix processes the data and returns an HTTP response where the <soap:Body> element describes whether or not the action was successful, and the details of any failure that might have occurred. As with all HTTP requests, the external client is blocked until it receives the response from the Web service.

See related topics on the format of the requests.

Integration scenario

The following diagram and integration scenario describe a typical Web service work flow for both the operational and querying service messages.



Operational and querying message web service

1. The client running on the external system composes a SOAP request message for the request Web service running on the Maintenix server machine. See example in the "SOAP request message structure" topic.

Note The thread in the external client is blocked until it receives the SOAP response.

2. The request Web service receives the SOAP request, and then invokes a method on an Enterprise Java Bean (EJB) in the Maintenix server.
3. The Maintenix server EJB processes the request, saves the results to the Maintenix database, and then returns the results to the request Web service.
4. The request Web service composes a SOAP response that contains the information requested by the external client (or a failure indication), and then sends it to the external client. The content of this request is specific to the event that occurred, and is documented in the associated Maintenix adapter guide. See example in the "SOAP response message structure" topic.

SOAP request message structure

The request Web service is the entry point into Maintenix for all operational and querying service messages.

Regardless of which adapter message is being called, external systems always call the single, generic SOAP-based "RequestWS" URL: `http://<username>:<password>@<servername>:<port>/mxcorews/services/RequestWS`

As specified in the Request.wsdl definition, the SOAP request message must include the <mode> element within the <Header> element of the message. The <mode> must be set to `sync`, which is used to indicate that the service must be processed, and that a response must be generated immediately.

You must also include a <request> element within the <Body> element of the SOAP message. Within the <request> element, you must enter XML-structured content that is based on the XSD of the adapter message that you want to invoke. Maintenix will look at the contents within the <request> element, and call the proper service based on the element name and namespace.

You can find the specific XSD files as well as sample messages for the content of the <request> element in the Maintenix download directory, `am-docs\Integrators and Developers` folder.

Note The sample messages included in the documentation folder do not contain the SOAP envelope.

The following is a sample SOAP request message that an external system might generate for an operational or querying message:

```
<?xml version='1.0' ?>
<soapenv:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
```

```
<soapenv:Header>
    <head:mode xmlns:head="http://xml.mxi.com/xsd/integration/
header/1.1">sync</head:mode>
</soapenv:Header>

<soapenv:Body>
    <req:request xmlns:req="http://xml.mxi.com/xsd/integration/
request/1.1">
        <!--The contents of the request message begins here.-->
        <!--The following example is taken from the aircraft-status-
request message-->
        <aircraft-status-request
          xmlns="http://xml.mxi.com/xsd/core/flight/aircraft-status-
request/1.0"
          xmlns:dt="http://xml.mxi.com/xsd/core/mxdomaintypes/2.0">
            <aircraft-identifier>
                <dt:barcode>I0005QVM</dt:barcode>
            </aircraft-identifier>
        </aircraft-status-request>
        <!--The contents of the request message ends here.-->
    </req:request>
   </soapenv:Body>
</soapenv:Envelope>
```

SOAP response message structure

Once the service is complete, Maintenix will return a SOAP-based XML message to the external system. This response message always includes a <response> element within the <Body> element of the SOAP message.

The <response> element has the following attributes:

- msg_id: The Maintenix-generated ID used to identify the particular message. It can be used for troubleshooting.
- msg_status: This is set to COMPLETE if the service was successful, or ERROR if it failed.

If the service is successful, then the <response> element contains an XML structure that is based on the XSD of the adapter message that was invoked. You can find the specific XSD files as well as sample messages for the content of the <request> element in the Maintenix download directory, am-docs\Integrators and Developers folder.

Note The sample messages included in the documentation folder do not contain the SOAP envelope.

The following is a sample SOAP response message that Maintenix might generate in response to an operational or querying request message:

```
<?xml version='1.0' ?>
<soapenv:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
```

```
   <soapenv:Body>
      <resp:response xmlns:resp="http://xml.mxi.com/xsd/integration/
response/1.1"
       msg_id="100024"
       msg_status="COMPLETE">
         <!--The contents of the response message begins here.-->
         <!--The following example is taken from the flight-
identifier-response message-->

         <flight-identifier-response
          xmlns="http://xml.mxi.com/xsd/core/flight/flight-identifier-
response/1.0">
            <external-key>0101BNAATL19990101</external-key>
            <flight-id>
               <flight-name>Duty</flight-name>
               <scheduled-departure-date>1999-01-01 03:38:00</
scheduled-departure-date>
               <departure-airport>ATL</departure-airport>
            </flight-id>
            <internal-identifier>4650:63699</internal-identifier>
         </flight-identifier-response>

         <!--The contents of the response message ends here.-->
      </resp:response>
   </soapenv:Body>

</soapenv:Envelope>
```

If the service is unsuccessful and there are errors, the <response> element contains an XML structure that describes the error. The XML message always includes an <errors> element within the <response> element. The <errors> element contains one or more <error> elements. Each <error> element describes the details of the failure that was encountered. Each <error> element contains the following child elements:

- <message>: A description of the problem that was encountered.
- <stacktrace>: A detailed trace of the exact line of code within Maintenix that generated the error. It is used for troubleshooting when communicating with your product support representative.

The following is a sample SOAP response message that Maintenix might generate in response to an unsuccessful operational or querying request message:

```
<?xml version='1.0' ?>
<soapenv:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

   <soapenv:Body>
      <resp:response
       msg_id="100031"
       msg_status="ERROR"
       xmlns:resp="http://xml.mxi.com/xsd/integration/response/1.1">
```

```
         <ns1:errors
          xmlns="http://xml.mxi.com/xsd/integration/error/1.1"
          xmlns:ns1="http://xml.mxi.com/xsd/integration/error/1.1"

 class="com.mxi.mx.core.adapter.flight.messages.flight.publish.
PublishingException"
          key="unknown"
          msg_id="100031">

            <ns1:error>
               <ns1:message>Unable to find valid inventory</
ns1:message>
               <ns1:stacktrace>
                  <!
[CDATA[<com.mxi.mx.core.adapter.flight.messages.flight.publish
                     .PublishingException: Unable to find valid
 inventory for aircraft
                     at
 com.mxi.mx.core.adapter.flight.messages.flight.publish

 .FlightPublishingService.generateAircraftStatus(Unknown Source)
                     at
 sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
                     ]]>

               </ns1:stacktrace>
            </ns1:error>
         </ns1:errors>
      </resp:response>
   </soapenv:Body>
</soapenv:Envelope>
```

Notification service messages

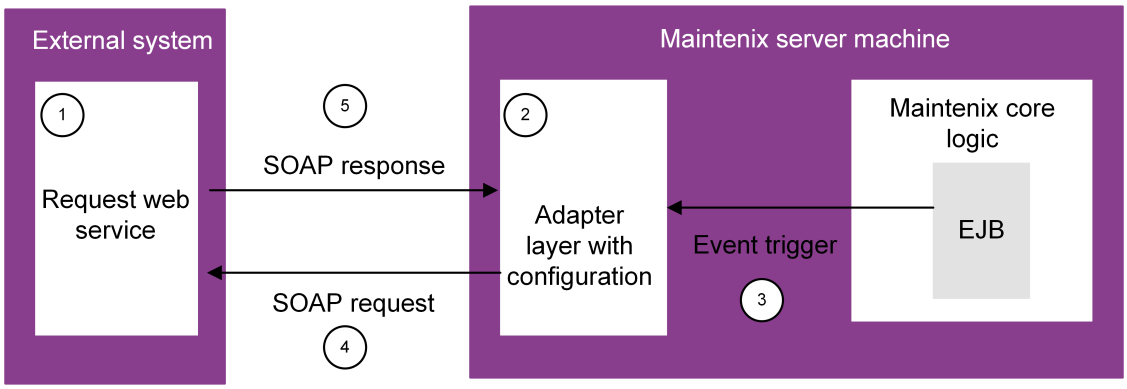
Notification service messages are used to alert an external system that something happened within Maintenix. For example an external system can be notified whenever a fault is raised, or a purchase order is updated.

To set up the notification service messages, the external system must expose an HTTP Web service for Maintenix to call. You must also configure Maintenix, providing the external system's Web service details (URL, username, password, etc.), as well as the message that the external system is expecting (namespace, root_name). When the specified event occurs within Maintenix, Maintenix will generate a request to the external system's Web service, providing the message information.

The external system's Web service can provide a response to Maintenix's HTTP request. Maintenix will acknowledge the response message, but there is no logic that occurs within Maintenix, regardless of whether the external system processed the message successfully or not.

Integration scenario

The following diagram and integration scenario describe a typical notification message Web service workflow:



Notification message web service

1. A Web service is developed on the client that can listen to the HTTP requests generated by Maintenix. You can build this Web service using any tool or programming language, as long as it can listen to HTTP requests.
2. Maintenix is configured for the destination Web services on the external system.

> **Note** When configuring Maintenix, the NAMESPACE and ROOT_NAME depend on the event that is being monitored, and is documented in the associated Maintenix adapter guide.

3. The triggering event occurs within Maintenix.
4. Maintenix composes a SOAP request message for the external system's Web service. The content of this request is specific to the event that occurred, and is documented in the associated Maintenix adapter guide.
5. The external system provides a response message, which is acknowledged by Maintenix.

Configuring Maintenix for the external system's web service

To configure the notification service, you must add a new row into Maintenix's INT_DEST_LOOKUP table for each separate instance of a notification messaging service. In the INT_DEST_LOOKUP table, you specify the external system's Web service details (URL, username, password, etc.), as well as the message that the external system is expecting (namespace, root_name).

You have the option to create multiple rows for the same notification message service. This enables you to send multiple messages each time a single triggering event occurs. For example,

you can have multiple external systems (different URL/USERNAME/PASSWORD values), or you can receive different XSD versions for the same triggering event (different NAMESPACE values).

The following table describes each column that you must populate in the INT_DEST_LOOKUP table.

Column	Data Type	Mandatory	Description
SUBSCRIPTION_DB_ID	Integer	Yes	This is the database identifier of this entry. This should be set to the DB_ID provided at time of database creation.
SUBSCRIPTION_ID	Integer	Yes	This is a unique value used to identify this configuration entry.
NAMESPACE	Char(100)	Yes	This is the namespace of the notification message's XML Schema (XSD). It is used to identify which schema will be generated when the triggering event occurs. You can find the name of the notification message in the Notification Service Messages section of the Maintenix Adapter guide. For example: http://xml.mxi.com/xsd/core/ flight/fault-updated-event/1.0.
ROOT_NAME	Char(100)	Yes	This is the name of the notification message. It is used to identify which message the external system is listening for. You can find the name of the notification message in the Notification Service Messages section of the Maintenix Adapter guide. For example: fault-updated-event.
URL	Char(100)	Yes	This is the URL of the external system's Web service. For example: http:// www.client.com:8080/service.
USERNAME	Char(80)	No	This is the username that Maintenix uses to login to the external system's Web service.
PASSWORD	Char(100)	No	This is the password that Maintenix uses to login to the external system's Web service.
HIST_BOOL	1 or 0	No	This value is always set to 0.

Column	Data Type	Mandatory	Description
STYLESHEET_PATH	Char(100)	No	This value is not needed and can be left empty.
DESTINATION	Char(20)	Yes	This value is always set to WS.

SOAP request message structure - notification service

When a triggering event occurs within Maintenix, Maintenix generates a SOAP-based XML message, and then sends it to the external system's Web service URL. The XML message always includes a <response> element within the <Body> element of the SOAP message.

The <response> element has the following attributes:

- msg_id: The Maintenix-generated ID used to identify the particular message. It can be used for troubleshooting.
- msg_status: This is set to COMPLETE

The <response> element contains an XML structure based on the XML Schema (XSD) of the notification message, as defined by the NAMESPACE specified in Maintenix's INT_DEST_LOOKUP table. You can find the specific XSD files, as well as sample messages for the content of the <response> element, in the Maintenix download directory, am-docs\Integrators and Developers folder.

Note The sample messages included in the documentation folder do not contain the SOAP envelope.

The following is a sample SOAP request message that Maintenix might generate for a notification message:

```
<?xml version='1.0' ?>
<soapenv:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

   <soapenv:Body>
      <resp:response xmlns:resp="http://xml.mxi.com/xsd/integration/
response/1.1"
       msg_id="100024"
       msg_status="COMPLETE">

         <!--The contents of the message begins here.-->
         <!--The following example is taken from the work-package-
updated-event message-->

         <work-package
```

```
          xmlns="http://xml.mxi.com/xsd/core/flight/work-package-
updated-event/1.0"
          xmlns:dt="http://xml.mxi.com/xsd/core/mxdomaintypes/2.0">
            <work-package-no>WO-T0000LDD</work-package-no>
            <name>A-CHECK (A Check)</name>
            <aircraft>
               <aircraft-identifier>
                  <dt:barcode>I0005QVM</dt:barcode>
                  <inventory-id>
                     <dt:oem-serial-number>203</dt:oem-serial-number>
                     <dt:oem-part-number>
                        <dt:oem-part-number>ACME505</dt:oem-part-
number>
                        <dt:manufacturer-id>
                           <dt:manufacturer-name>Acme</
dt:manufacturer-name>
                           <dt:manufacturer-code>ABI</dt:manufacturer-
code>
                        </dt:manufacturer-id>
                     </dt:oem-part-number>
                  </inventory-id>
                  <dt:aircraft-identifier>CFDRH</dt:aircraft-
identifier>
               </aircraft-identifier>
               <aircraft-status>
                  <capability />
                  <operating-status>In Repair</operating-status>
               </aircraft-status>
            </aircraft>
            <barcode>T0000LDD</barcode>
            <status />
            <work-type />
            <type-subtype>CHECK</type-subtype>
            <location>
               <airport-code>ORD</airport-code>
               <scheduled-location-code>ORD/HGR-2/TRK-4</scheduled-
location-code>
               <scheduled-location-name>Track-4</scheduled-location-
name>
               <scheduled-location-type>TRACK</scheduled-location-
type>
            </location>
            <scheduled-date>
               <start-date>2002-04-13 06:30:50</start-date>
               <end-date>2002-04-13 02:30:47</end-date>
            </scheduled-date>
            <actual-date />
            <external-vendor>
               <vendor-name />
               <vendor-code />
            </external-vendor>
            <delay />
         </work-package>

         <!--The contents of the message ends here.-->
```

```
      </resp:response>
   </soapenv:Body>

</soapenv:Envelope>
```

SOAP response message structure - notifications

The external system's Web service can provide a response to Maintenix's HTTP request. Maintenix will then acknowledge the response message, but there is no logic that occurs within Maintenix, regardless of whether the external system processed the message successfully or not.

The following is a sample SOAP response message that an external system might generate for a notification message:

```
<?xml version="1.0" encoding="UTF-8"?>
<en:Envelope xmlns:en="http://schemas.xmlsoap.org/soap/envelope/"/>
```

Setting up the IFS Maintenix Message Ordering Processor

The IFS Maintenix Message Ordering Processor provides message-ordering capabilities at the application layer. This means that applications which use Maintenix APIs are responsible for proper message ordering.

Note If the external system is unable to add sequence numbers to its messages, then message ordering is not possible using the IFS Maintenix Integration Framework.

Message streams

The IFS Maintenix Message Ordering Processor supports multiple message streams. A message stream is a logical sequence of messages which must be processed in a specified order. Messages in a particular stream need to be processed in relation to other messages in a stream, but not in relation to messages outside of the stream. Note that not all messages necessarily belong to a message stream.

Adding new message streams

Before ordered messages can be sent to a message stream in the Core Transaction Engine, a row describing the message stream must be added to the INT_MESSAGE_ORDER table. Create a row with the following information:

Parameters to be set in the INT_MESSAGE_ORDER

Column Name	Column Type	Description
stream_cd	VARCHAR2(16)	Primary key that uniquely represents the ordered message stream.
stream_sdesc	VARCHAR2(80)	Short description of the ordered message stream.
seq_number_last_processed	NUMBER(20)	Last sequence number that was processed in this ordered message stream. Initialize this value to 0
seq_number_next	NUMBER(20)	Next sequence number that should be processed in the ordered message stream. Initialize this value to 1

Configuring message order timing

In the UTL_CONFIG_PARM table, the MESSAGE_ORDER_SLEEP_TIME and MESSAGE_ORDER_TIMEOUT configuration parameters control the timing associated with message ordering.

Note The MESSAGE_ORDER_SLEEP_TIME and MESSAGE_ORDER_TIMEOUT configuration parameters shared by all message streams created in an instance of Maintenix.

Configuration Parameters

Configuration Parameter	Configuration Type	Description
MESSAGE_ORDER_TIMEOUT	INTEGRATION	Maximum amount of time (in seconds) that the IFS Maintenix Message Ordering Processor will wait before skipping the next expected message.

Configuration Parameter	Configuration Type	Description
MESSAGE_ORDER_ SLEEP_TIME	INTEGRATION	The amount of time (in milliseconds) that the IFS Maintenix Message Ordering Processor will wait before repeating its cycle of acquiring a database lock.

The MESSAGE_ORDER_SLEEP_TIME configuration parameter allows the message ordering to be throttled as necessary. A higher sleep time value will reduce the frequency of queries to the database, but will also reduce the processing speed of ordered messages. The value should be tuned appropriately for the specific solution required by the customer.

If an ordered message is delayed in its delivery from an external system, processing of further messages from the ordered stream is halted until the missing message is delivered. However, without intervention, a message that is never delivered due to corruption or some other error emanating from the external system could permanently halt the stream. In order to avoid this, the MESSAGE_ORDER_TIMEOUT configuration parameter specifies a timeout interval for the missing message to arrive. This will ensure that message processing will never cease to function completely due to missing messages.

Note If an expected message appears only after the timeout has expired, and appears after another message that had a higher sequence number, then it will never be processed. If the message must still be processed in this scenario, it will have to be processed manually or resubmitted with a higher sequence number.

Sequencing inbound messages

In order for a message to be processed by the IFS Maintenix Message Ordering Processor, it needs to specify the message stream to which it belongs, as well as its sequence number in the stream.

The following variables should be included in the SOAP message or JMS properties:

Message Property	Property Type	Description
msg_stream	VARCHAR2(16)	The name of the message stream you would like to send the message to; stream_cd from the INT_MESSAGE_ORDER table

Message Property	Property Type	Description
msg_id	NUMBER(20)	The sequence number of the message; the message will be processed when this property matches seq_number_next in the INT_MESSAGE_ORDER table [minimum of 5 characters]

In addition, the following should be included as a JMS property:

Message Property	Property Type	Description
MX_SEQUENTIAL	Boolean	Indicates that the message is part of a message stream; should be set to `true` for all ordered messages

Managing integration messages in Maintenix

Maintenix supports the searching, viewing, and archiving of all Maintenix Adapter messages. You can also send individual messages through the Maintenix GUI to help test and debug your messaging integration.

Viewing integration messages

You can view all integration messages in the Maintenix GUI. Use the Message Search page to retrieve a list of messages. Use the Message Details page to view specific information about a message.

Use the Message Search page to return a list of messages. For each message that is returned in your search, Maintenix provides the following information:

Message element	Description
Queue ID	This number is generated by Maintenix.
Message ID	This number or code is the unique identifier of the message. The Message ID may be specified in the message itself, or else it is generated by Maintenix automatically. Click the Message ID hyperlink to go to the Message Details page, where you can find specific information about the message.

Message element	Description
Stream	The code of the stream to which the message belongs (applicable to Message Ordering Framework). This is null if the message does not belong to a stream.
Status	The status of the message sent to Maintenix. The status can be one of the following: • AWAIT: The message is still processing. • ERROR: The message encountered an error while processing. • SUCCESS: The message was processed successfully.
Message Type	The type of integration framework message.
Start Time	The time that the message is received and processing begins.
Archived	If this check box is selected, the message is archived.

From the Message Search page, you can link to message details pages that provide additional information including the following:

Inbound tab

Item	Description
Delivery Code	The restrictions on message processing. The delivery code can be one of the following: • DEFAULT: There are no restrictions on message processing. • FIFO: The message must be processed in sequence by the order received. • ORDERED: The message must be processed through the Message Ordering Processor.
Message Source	The technology by which an external system sent the message to Maintenix. The message source code can be one of the following: • ws: The message was received through a web service. • jms: The message was received through a JMS queue.
Root Namespace	The namespace of the root element of the XML message.
Message	The contents of the XML message.

Outbound tab

Item	Description
Root Name	The name of the root element of the XML message.
Root Namespace	The namespace of the root element of the XML message.
Message	The contents of the XML message.

Errors tab: The Errors tab appears for messages that contain errors.

Results tab:When a message is processed successfully, the Results tab lists the details for any of the following actions that are performed by Maintenix:

- Creating a work package.
- Updating a work package.
- Creating a task.
- Updating a task.
- Creating a fault.

Note Maintenix uses the Alerts framework to populate the Results tab. The Alert Engine must be active, and the EMA alert types must be enabled for the Results tab to be populated.

Notification tab: The Notification tab lists all notification log entries related to the integration message. All notifications sent to a subscriber through the event listener service are listed here.

Process tab: The Process tab lists all process log entries related to the integration message.

Sending messages manually within Maintenix

You can send integration messages to Maintenix manually through the Maintenix GUI as if they are being sent from an actual external system.

Use the Echo and Sleep functionalities when sending a message through the Maintenix GUI to test the framework. The Echo function returns a response immediately, while the Sleep function waits for the number of seconds specified in the message before returning a response.

Note "xmlns" is the default name space used by the parser to retrieve objects (elements) and values from the request message. If the request message contains elements that are defined in other name spaces, these XML elements must be prefixed with the name space. Otherwise, the parser cannot retrieve the element value correctly, and Maintenix business logic may fail.

If the request message declares a namespace URL as xmlns:mxdomain="http://xml.mxi.com/xsd/core/mxdomaintypes/1.0, then elements defined in xmlns:mxdomain should be encoded as illustrated in the following example:

```
<mxdomain:contact>
    <mxdomain:username>jdoe</mxdomain:username>
</mxdomain:contact>
```

Send message manually

To test and debug your integration, you can send an XML message manually from within Maintenix. Maintenix processes the message and provides details, including any errors that occurred, on the Message Details page.

Prerequisite

You must have the ACTION_SEND_MESSAGE permission enabled.

Steps

1. Select the **Send Integration Message** menu item.
2. On the **Send Message** page, provide the following information:
   - Choose one of the following:
     - **Asynchronous**: If this check box is selected, the message will be processed asynchronously. Asynchronous operation mode is recommended for requests that may require more than 60 seconds of processing by the Maintenix server.
     - **Sequential**: If this check box is selected, the messages are processed in order, such that the processing of the next sequential message will occur only after the previous sequential message has been processed.
   - **Transport**: The manner in which the message will be sent to Maintenix. Choose one of the following:
     - Web Service: The message will be sent to the Maintenix request Web service.
     - Remote Method Invocation: The integration framework will be invoked directly from within Maintenix, bypassing the Web service; when asynchronous, the message will be placed directly on the inbound integration queue, and when synchronous, the message will be processed immediately.
   - **Base URL**: The URL that specifies the server from which the message will be sent; this field is read-only and will always identify the server to which you are currently logged in.
   - **Username**: Enter your username.
   - **Password**: Enter your password.
   - **Sample Message**: If you are testing the framework, click either the Echo button or the Sleep button to populate the Body field with a pre-formatted test message.

- **Body**: Enter your XML message. For information on the message formatting and schema details, refer to the appropriate Maintenix API guide.

3. Click **OK**.

Edit and resend an integration message

If you encounter an error when manually sending an integration message into Maintenix, you can correct the error in your XML and resend it.

Prerequisite

You must have the ACTION_RESEND_MESSAGE permission.

Steps

1. Go to the **Message Details** page.
2. Click **Edit and Resend**.
3. On the **Send Message** page, edit the XML message.
4. Click **OK**.

Understanding message errors

When a message is sent, the Message Details page returns a list of any errors on the Error tab. You can use the error details to troubleshoot the error.

Maintenix tracks the following types of errors:

- Integration framework errors
- Business logic errors

An integration framework error results from an improperly formatted message. Such errors can include XML tagging errors, namespace destination errors, and errors resulting from unlicensed namespace references.

When an integration error occurs, Maintenix displays a description of the error, and a stack trace of the Maintenix component that detected the error.

A business logic error is an error in the data contained in the XML message. Business logic errors include references to entities that do not exist in Maintenix, such as unknown locations, or unknown part numbers.

When a business logic error occurs, Maintenix displays a description of the error and an error element path that identifies the element that contained the problematic data.

Archive integration messages

You cannot delete integration message records through the Maintenix GUI. To facilitate message searching, old integration messages can be archived. You can then filter out archived messages from your message searches.

Prerequisite

You must have the ACTION_ARCHIVE_MESSAGE permission enabled.

Steps

1. Go to the **Message Search** page.
2. Enter search criteria and click **Search**.
3. In the **Messages Found** list, select one or more messages and click **Archive**.

Logging

Integration events, exceptions, and responses are logged in the Maintenix log files. The log files use standard formats that you can configure to work with your own log processing and monitoring tools.

The primary log file is `maintenix.log`. This file uses Log4j format.

HTTP requests are also logged in `weblogic-access.log` which you have the option to monitor in real time.

For information on configuring logging, see the *Maintenix Installation and Configuration Guide*.

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.