IFS Aviation Maintenance

Finance 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 Finance API....8
Purchasing and invoicing integrations with an external system....8
Notifying external systems of significant financial activities....9

2 Setting up the Finance API....10
License requirement....10
Permissions and authentication....11
Assign the Finance integration role to a user....11
Give Finance integration permission to a user role....11
Configuration parameters....12
Alerts....12

3 Integration framework....14
Operational and querying service messages....14
SOAP request message structure....16
SOAP response message structure....17
Notification service messages....19
Configuring Maintenix for the external system's web service....20
SOAP request message structure - notification service....22
SOAP response message structure - notifications....24
Setting up the IFS Maintenix Message Ordering Processor....24
Message streams....24
Adding new message streams....24
Configuring message order timing....25
Sequencing inbound messages....26
Managing integration messages in Maintenix....27
Viewing integration messages....27
Sending messages manually within Maintenix....29
Send message manually....30
Edit and resend an integration message....31
Understanding message errors....31
Archive integration messages....32
Logging....32

4 Finance API scenarios....33
Purchasing and invoicing integration scenarios....33
Performing invoice matching in external finance systems....33
Performing invoice matching in Maintenix....35
Operational service scenarios....35
Updating the currency exchange rate....36
Creating a financial account....36
Creating a financial sub-account....36
Creating a financial account hierarchy....37
Updating a financial account....37
Closing a financial account....38
Opening a financial account....38
Approving a budget order check....39
Rejecting a budget order check....39
Querying service scenarios....40
Requesting a detailed financial log for a financial period....40
Requesting a detailed financial log for a transaction....40
Requesting a summary of account transactions for a financial period....41
Requesting all invoices marked "to be paid" since a specified date....41
Requesting a particular invoice....41
Requesting all orders modified since a specified date....42
Requesting a particular order....42
Notification service scenarios....42
Publishing a detailed financial log automatically....43
Publishing an invoice marked as "to be paid" automatically....43
Publishing order modifications automatically....43
Publishing an order inventory received and inspected as serviceable automatically....44
Publishing an order inventory returned to vendor automatically....44

5 Operational service messages....46
create-purchase-invoice-4.0....46
Sample messages....47
Message contents....47
Mandatory elements....47
Errors and exceptions....51
Mapping elements to Maintenix....51

purchase-invoice-created-1.1 53
mark_invoice_as_paid-2.0 53
Sample messages 53
Message contents 53
Mandatory elements 54
Errors and exceptions 54
Mapping elements to Maintenix 54
invoice_paid-1.0 55
update_currency_exchange_rate-2.0 55
Sample messages 55
Message contents 55
Mandatory elements 56
Errors and exceptions 56
Mapping elements to Maintenix 56
currency_exchange_rate_updated-1.0 56
create-financial-accounts-request-1.0 57
Sample messages 57
Message contents 57
Mandatory elements 57
Errors and exceptions 58
Mapping elements to Maintenix 58
create-financial-accounts-response-1.0 59
update-financial-accounts-request-1.0 59
Sample messages 59
Message contents 59
Mandatory elements 60
Errors and exceptions 61
Mapping elements to Maintenix 61
update-financial-accounts-response-1.0 61
close-financial-accounts-request-1.0 61
Sample messages 62
Message contents 62
Mandatory elements 62
Errors and exceptions 62
Mapping elements to Maintenix 63
close-financial-accounts-response-1.0 63
open-financial-accounts-request-1.0 63

Sample messages....63
Message contents....63
Mandatory elements....64
Errors and exceptions....64
Mapping elements to Maintenix....64
open-financial-accounts-response-1.0....65
approve_order_budget-2.0....65
Sample messages....65
Message contents....65
Mandatory elements....65
Errors and exceptions....66
Mapping elements to Maintenix....66
reject_order_budget-2.0....66
Sample messages....66
Message contents....67
Mandatory elements....67
Errors and exceptions....67
Mapping elements to Maintenix....67

6 Querying service messages....68
request-detailed-inventory-financial-log-3.0....68
Sample messages....68
Message contents....69
request-detailed-inventory-financial-log-by-id-3.0....69
Sample messages....69
Message contents....69
request_summary_inventory_financial_log-2.0....69
Sample messages....69
Message contents....70
send_summary_inventory_financial_log-2.0....70
Sample messages....70
Message contents....70
Mapping elements to the Maintenix GUI....70
request_to_be_paid_invoices-2.0....71
Sample messages....71
Message contents....71
request_to_be_paid_invoices_by_id-2.0....72

Sample messages....72
Message contents....72
request-order-information-4.0....72
Sample messages....72
Message contents....73
request_order-information-by-id-4.0....73
Sample messages....73
Message contents....73

7 Notification Service Messages....74
send-detailed-inventory-financial-log-3.0....74
Sample messages....75
Message triggers....75
Mapping elements to the Maintenix GUI....76
send-order-information-4.0....76
Sample messages....77
Message triggers....77
Mapping elements to the Maintenix GUI....78
send_to_be_paid_invoices-2.0....80
Sample messages....80
Message triggers....81
Mapping elements to the Maintenix GUI....81
order-inventory-received-1.0....82
Sample messages....83
Message triggers....83
Mapping elements to the Maintenix GUI....83
order-inventory-returned-1.0....84
Sample messages....85
Message triggers....85
Mapping elements to the Maintenix GUI....85

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 Finance API

The Finance API integrates Maintenix with an external financial system for customers using Maintenix for materials. When using Maintenix for materials, inventory-related financial information is generated within Maintenix, and this information is sent to the external financial system, using the Finance API. Also, the Finance API synchronizes Maintenix's purchasing and invoicing information with an external system that handles the accounts payable.

The following high-level scenarios describe the supported situations where you can use the Finance API to integrate your external system with Maintenix:

- Purchasing and invoicing integrations with an external system.
- Notifying external systems of significant financial activities.

Purchasing and invoicing integrations with an external system

In order for Maintenix materials to function correctly, Maintenix must track purchase orders, invoices, and material receipts.

Matching purchase orders to invoices and materials receipt is called invoice matching, or three-way matching, and it is necessary for Maintenix to accurately monitor inventory and record inventory-related financial information. However, Maintenix is not an accounts payable system; Maintenix cannot track vendor payments and banking transactions. The purchasing and invoicing information must be shared between Maintenix and the external financial system.

Maintenix is always the purchasing system, and the external system is always the payment system; therefore, the invoice matching can be completed in either system. The Finance API can then share that information with the other system.

When invoice matching is performed within Maintenix, all purchase orders are created and tracked in Maintenix, all material receipts are recorded in Maintenix, and all invoices are created and matched to orders in Maintenix. When the order is completed and the material is received, the invoice is validated for payment. At this point, you can use the Finance API to send all invoices validated for payment (marked "to be paid") to the external system.

When the invoice matching is performed by the external system, the orders are still created and tracked within Maintenix, but the external system must also have a duplicate set of orders so that it can create and match the invoices. Every time an order is created or updated in any way, you can use the Finance API to send this information to the external system. The external system then performs the invoice creation and matching. Once the invoice is validated for payment in the external system, you can use the Finance API to create the corresponding completed invoices in Maintenix.

Notifying external systems of significant financial activities

You can use the notification or querying services provided with the Finance API to report significant information to external systems such as the financial transaction logs, and order or invoice information.

When using the querying services, the external system sends a message requesting information, and Maintenix returns that information to the external system. When using the notification services, Maintenix automatically sends messages to the external system upon specified events.

2

Setting up the Finance API

Most messages in the Finance API are sent using a web service or JMS framework that integrates directly with Maintenix. See *Integration framework*.

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

- `open-financial-accounts-request-1.0`
- `open-financial-accounts-response-1.0`

For the Open financial accounts messages, the WSDL is available at: `http://<maintenix_server>:<http_port>/finance/AccountService?wsdl`

The request API endpoint is `http://<maintenix_server>:<http_port>/finance/AccountService`

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

Note For existing customers, that have the IFS Maintenix Integration Framework endpoints configured for this sub-set of messages, you must use the specified web service, but the IFS Maintenix Integration Framework endpoints are still valid; request messages sent to the IFS Maintenix Integration Framework endpoints are rerouted to the http://<maintenix_server>:<http_port>/finance/AccountService endpoint.

License requirement

To use the Finance API, you must have the `FINANCE_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 `FINANCE_ADAPTER` are installed, a default user called `fncintegration` with the FNCINTEG role is automatically created. The FNCINTEG role, gives the `fncintegration` user permission to authenticate and authorize inbound messages from the Finance API to Maintenix. Any other user created for this purpose should be assigned the FNCINTEG role.

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

By default, the FNCINTEG role is assigned the following permission:

- API_OPEN_FINANCE_ACCOUNT_REQUEST

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

Assign the Finance integration role to a user

You can grant specific users permission to use the Finance API.

Steps

1. Go to the **Role Search** page.
2. Enter FNCINTEG 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**. Otherwise, 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 Finance integration permission to a user role

You can grant users, with a specific role, permission to use the Finance 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 permission name "API_OPEN_FINANCE_ACCOUNT_REQUEST" in the **Param Name** field and click **Search**.
6. Select the permission check box and click **Enable**.
7. Click **OK**.

Configuration parameters

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

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.

Alerts

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

Finance API alerts

Alert name	Description	Message
Order Inventory Returned Send Failure	This alert is raised if something goes wrong while publishing the Order Inventory Returned message.	Order Inventory Returned message for the shipment [{shipment ID}] could not be published. Error(s): [error code]
Order Inventory Received Send Failure	This alert is raised if something goes wrong while publishing the Order Inventory Received message.	Order Inventory Received message for the inventory [{inventory ID}] could not be published. Error(s): [error code]

3

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`

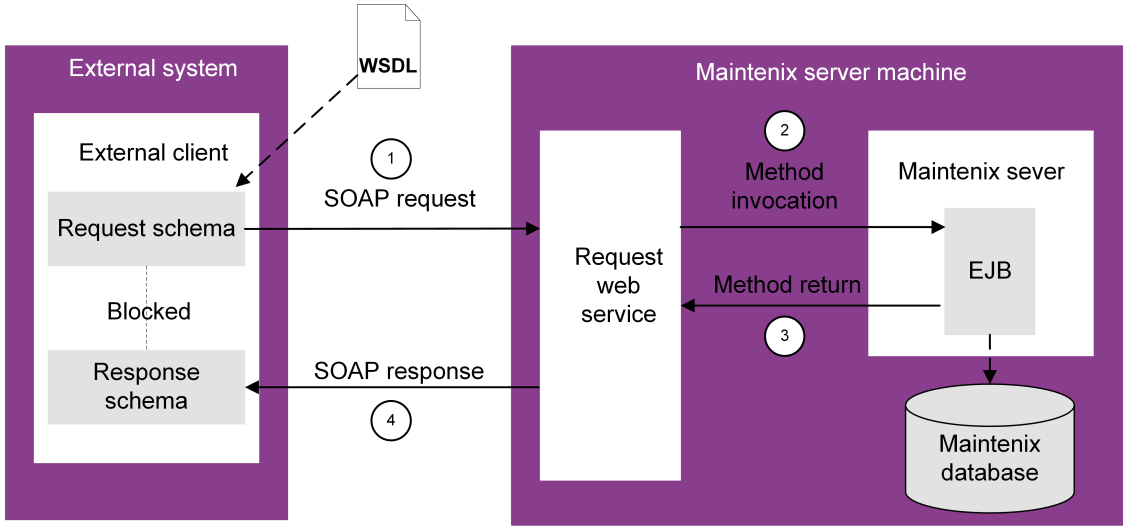
The same Web service is called, regardless of the data that is passed.

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

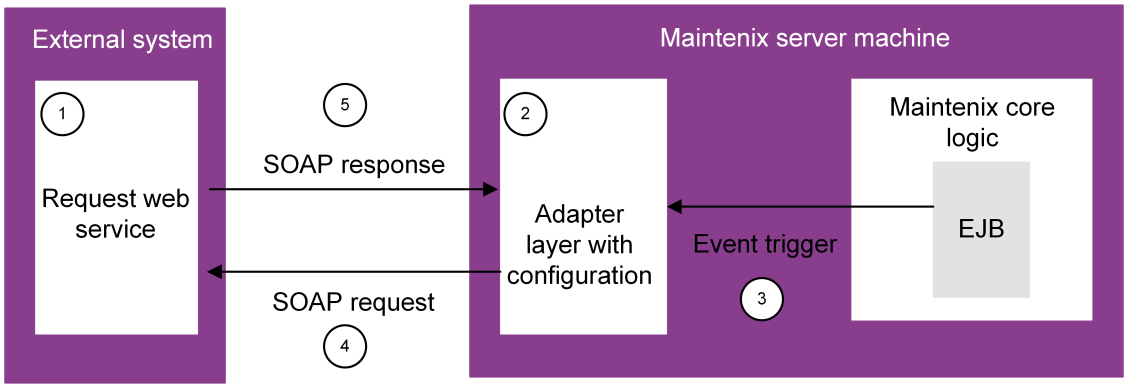
Notification service messages are used to alert an external system that something happened within Maintenix. For example an external system can be notified whenever a fault is raised, or a purchase order is updated.

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*.

4

Finance API scenarios

Purchasing and invoicing integration scenarios

The Finance API supports the following purchasing and invoicing integration scenarios:

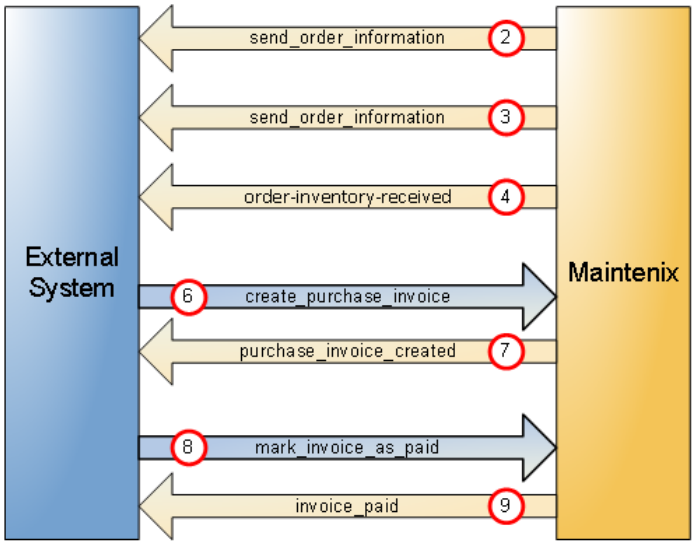
- Performing invoice matching in external finance systems.
- Performing invoice matching in Maintenix.

Performing invoice matching in external finance systems

Goal: To prepare orders and perform inventory receipts in Maintenix, but perform invoice matching and accounts payable in the external finance system.

Summary: Purchasing and receiving is always done in Maintenix, while accounts payable is always done in the finance system; however, invoice management and matching can be done in either system. This scenario describes what happens when the invoice management and matching is done in the finance system. The creation of all orders (purchase, repair, exchanges, and so on) and all receiving is done in Maintenix. Invoices are received, processed, and validated for payment (three-way matching) in the financial system. In order for this to work, the finance system needs to know about all orders and receipts that have occurred in Maintenix. Also, for Maintenix's inventory value calculations to work, Maintenix must know the final invoice price of every order.

Messaging: The following diagram and steps outline the messaging for this scenario.



Messaging for order matching in an external system

1. You create and update all orders in Maintenix. All inventory receipts occur within Maintenix.
2. Maintenix sends a *send-order-information-4.0* message to the external system when an order is created in Maintenix.
3. Maintenix sends a *send-order-information-4.0* message to the external system when an order is updated in Maintenix. This includes adding lines, adding or removing taxes, adding or removing charges, assigning or unassigning part requests, converting order lines, updating line prices, editing order details, and adding notes to an order. Maintenix also sends a *send-order-information-4.0* message to the external system when authorization is requested for an order, or an order is authorized, unauthorized, issued, acknowledged, or canceled.
4. Maintenix sends a *order-inventory-received-1.0* message to the external system when the inventory for an order is received in Maintenix.
5. You receive the invoice, compare it against the original order, compare it against receipts, and deal with any discrepancies as necessary.
6. The external system sends a *create-purchase-invoice-4.0* message to Maintenix when the invoice is three-way matched and approved for payment in the external system.
7. Maintenix sends a *purchase-invoice-created-1.1* message to the external system confirming that the invoice is created in Maintenix.
8. The external system sends a *mark_invoice_as_paid-2.0* message to Maintenix when the invoice is paid in the external system.
9. Maintenix returns an *invoice_paid-1.0* message to the external system, confirming that the invoice is marked as paid in Maintenix.

Result: Maintenix and the external system have identical copies of the final, revised, and approved order. Maintenix and the external system have identical copies of the final, revised, and approved invoice.

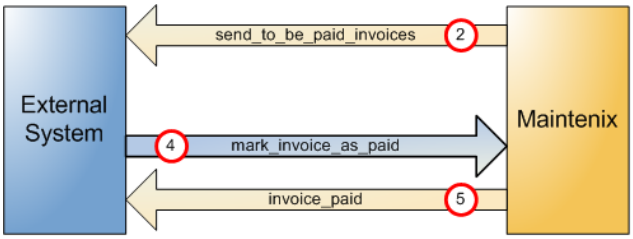
Performing invoice matching in Maintenix

Goal: To prepare orders, perform receipts, and perform invoice matching in Maintenix, but perform accounts payable in the external finance system.

Summary: Purchasing and receiving is always done in Maintenix, while accounts payable is always done in the finance system; however, invoice management and matching can be done in either system. This scenario describes what happens when the invoice management and matching is done in Maintenix.

The creation of all orders (purchase, repair, exchanges, and so on), and all receiving is done in Maintenix. In addition, invoices are received, processed, and validated for payment (three-way matching) in Maintenix. The finance system is informed of the final, validated invoices so that it can process them for payment (accounts payable). For this to work, the finance system needs to know about all invoices, when the three-way matching is complete.

Messaging: The following diagram and steps outline the messaging for this scenario.



Messaging for order matching in Maintenix

1. You create all orders, perform all receipts, enter all invoices, and perform three-way matching in Maintenix.
2. Maintenix sends a *send_to_be_paid_invoices-2.0* message to the external system when an invoice is validated for payment in Maintenix.
3. You pay the invoice (accounts payable) using the external system.
4. The external system sends a *mark_invoice_as_paid-2.0* message to Maintenix when the invoice is paid in the external system.
5. Maintenix returns an *invoice_paid-1.0* message to the external system confirming that the invoice is marked as paid in Maintenix.

Result: Maintenix and the external system have identical copies of the final, revised, and approved invoice.

Operational service scenarios

The Finance API supports the following operational service scenarios:

- Updating the currency exchange rate

- Creating, updating, opening and closing a financial account.
- Creating a financial sub-account.
- Creating a financial account hierarchy.
- Approving and rejecting a budget order check.

Updating the currency exchange rate

Goal: To update the currency exchange rate in Maintenix.

Summary: The external system sends a message to update the exchange rate of a currency in Maintenix. Preconditions:

- The currency exists in Maintenix.
- The new exchange rate is provided.

Messaging: The external system sends an inbound *update_currency_exchange_rate-2.0* message. Maintenix returns a *currency_exchange_rate_updated-1.0* outbound message.

Result: The value of the exchange rate in Maintenix is updated.

Creating a financial account

Goal: To create a new financial account in Maintenix.

Summary: When a financial account is created in an external system, the external system sends a message to create a corresponding account in Maintenix. This ensures that both the external system and Maintenix remain synchronized.

Preconditions:

- The account code does not exist in Maintenix.
- The external key code does not exist in Maintenix.
- The account type is defined in Maintenix already.
- The t-code is already defined in Maintenix.

Messaging: The external system sends an inbound *create-financial-accounts-request-1.0* message. Maintenix returns a *create-financial-accounts-response-1.0* outbound message.

Result: The new account is created in Maintenix.

Creating a financial sub-account

Goal: To create a new financial sub-account for an existing parent account in Maintenix.

Summary: When a financial sub-account is created in an external system, the external system sends a message to create a corresponding sub-account in Maintenix. This ensures that both the external system and Maintenix remain synchronized.

Preconditions:

- The account code does not exist in Maintenix.
- The existing account code of the parent account is provided.
- The external key code does not exist in Maintenix.
- The account type is defined in Maintenix already.
- The t-code is already defined in Maintenix.

Messaging: The external system sends an inbound *create-financial-accounts-request-1.0* message. Maintenix returns a *create-financial-accounts-response-1.0* outbound message.

Result: The new sub-account is created in Maintenix.

Creating a financial account hierarchy

Goal: To create a new financial account hierarchy in Maintenix.

Summary: A financial hierarchy is a tree-like arrangement of accounts and sub-accounts. When a financial account hierarchy is created in an external system, the external system sends a message to create a corresponding account hierarchy in Maintenix. This ensures that both the external system and Maintenix remain synchronized.

Preconditions:

- The account codes do not exist in Maintenix.
- The external key codes do not exist in Maintenix.
- The account types are defined in Maintenix already.
- The t-code is already defined in Maintenix.
- All sub accounts meet the same criteria as accounts.

Messaging: The external system sends an inbound *create-financial-accounts-request-1.0* message. Maintenix returns a *create-financial-accounts-response-1.0* outbound message.

Result: The new account hierarchy is created in Maintenix.

Updating a financial account

Goal: To update a financial account in Maintenix.

Summary: When a financial account is updated in an external system, the external system sends a message to update the corresponding account in Maintenix. This ensures that both the external system and Maintenix remain synchronized.

Preconditions:

- The account exists in Maintenix.

Messaging: The external system sends an inbound *update-financial-accounts-request-1.0* message. Maintenix returns an *update-financial-accounts-response-1.0* outbound message.

Result: The account is updated in Maintenix.

Closing a financial account

Goal: To close a financial account in Maintenix.

Summary: When a financial account is closed in an external system, the external system sends a message to close the corresponding account in Maintenix. This ensures that both the external system and Maintenix remain synchronized.

Preconditions:

- The account exists in Maintenix.
- The account is not a default account.
- The account does not contains open sub-accounts.
- The account is not assigned to aircraft with statuses other than condemned or scrapped.
- The account is not assigned to task definitions with statuses other than superseded or obsolete.
- The account is not assigned to a tax or charge.
- The account is not assigned to a part number.

Messaging: The external system sends an inbound *close-financial-accounts-request-1.0* message. Maintenix returns a *close-financial-accounts-response-1.0* outbound message.

Result: The account is closed in Maintenix.

Opening a financial account

Goal: To open a financial account in Maintenix.

Summary: When a financial account is opened in an external system, the external system sends a message to open the corresponding account in Maintenix. This ensures that both the external system and Maintenix remain synchronized.

Preconditions:

- The account exists in Maintenix.
- The account does not have a closed parent account.

Messaging: The external system sends an inbound *open-financial-accounts-request-1.0* message. Maintenix returns an *open-financial-accounts-response-1.0* outbound message.

Result: The account is opened in Maintenix.

Approving a budget order check

Goal: To approve the budget request for an order in Maintenix.

Summary: When the budget workflow is enabled in Maintenix, an order's budget must be approved before the order can be authorized. When a user requests authorization for an order in Maintenix, the Budget Check status is set to REQUESTED, and Maintenix notifies the external system. You must approve the order's budget from an external system using approve_order_budget message before you can authorize the order. You cannot approve the budget using the Maintenix UI.

Preconditions:

- The ENABLE_ORDER_BUDGET_INTEGRATION configuration parameter is enabled.
- The specified order exists in Maintenix.
- The budget check status for the order is set to REQUESTED.

Messaging: Maintenix publishes a *send-order-information-4.0* outbound message. The external system sends an inbound *approve_order_budget-2.0* message. Maintenix returns a *send-order-information-4.0* outbound message.

Result: The Budget Check status for the order in Maintenix is set to APPROVED.

Rejecting a budget order check

Goal: To reject the budget request for an order in Maintenix.

Summary: When the budget workflow is enabled in Maintenix, an order's budget must be approved before the order can be authorized. When a user requests authorization for an order in Maintenix, the Budget Check status is set to REQUESTED, and Maintenix notifies the external system. You can reject the order's budget from an external system using approve_order_budget message. You cannot reject the budget using the Maintenix UI.

Preconditions:

- The ENABLE_ORDER_BUDGET_INTEGRATION configuration parameter is enabled.
- The specified order exists in Maintenix.
- The budget check status for the order is set to REQUESTED.

Messaging: Maintenix publishes a *send-order-information-4.0* outbound message. The external system sends an inbound *reject_order_budget-2.0* message. Maintenix returns a *send-order-information-4.0* outbound message.

Result: The Budget Check status for the order in Maintenix is set to REJECTED.

Querying service scenarios

The Finance API supports the following querying service scenarios:

- Requesting a detailed financial log for a financial period.
- Requesting a detailed financial log for a transaction.
- Request a summary of account transactions for a financial period.
- Requesting all invoices marked "to be paid" since a specified date.
- Requesting a particular invoice.
- Requesting all orders modified since a specified date.
- Requesting a particular order.

Requesting a detailed financial log for a financial period

Goal: To provide the external system with a detailed financial log for a financial period.

Summary: The external system sends a message identifying the start and end date of the financial period. Maintenix returns a message containing all inventory-related financial transactions occurring within the specified financial period.

Preconditions:

- The start and end dates are provided in the message.

Messaging: The external system sends an inbound *request-detailed-inventory-financial-log-3.0* message. Maintenix returns a *send-detailed-inventory-financial-log-3.0* outbound message.

Result: Maintenix returns the detailed financial log for the financial period.

Requesting a detailed financial log for a transaction

Goal: To provide the external system with a detailed financial log for a particular transaction.

Summary: The external system sends a message identifying the ID of the transaction. Maintenix returns a message containing the financial log information.

Preconditions:

- The transaction ID is provided in the message.

Messaging: The external system sends an inbound *request-detailed-inventory-financial-log-by-id-3.0* message. Maintenix returns a *send-detailed-inventory-financial-log-3.0* outbound message.

Result: Maintenix returns the detailed financial log for the transaction.

Requesting a summary of account transactions for a financial period

Goal: To provide the external system with a summary of account transaction totals for a specific financial period.

Summary: The external system sends a message identifying the start and end dates of the financial period. Maintenix returns a message containing each account affected by inventory-related financial transactions within the time period specified, along with the total debit or credit to each account.

Preconditions:

- A start and end dates are provided.

Messaging: The external system sends an inbound *request_summary_inventory_financial_log-2.0* message. Maintenix returns a *send_summary_inventory_financial_log-2.0* outbound message.

Result: Maintenix returns the summary inventory financial log.

Requesting all invoices marked "to be paid" since a specified date

Goal: To provide the external system with the details of all Maintenix invoices that have been marked as "to be paid" since a specified date.

Summary: The external system sends a message identifying a date. Maintenix returns a message containing all invoices with a TOBEPAID status, that had been marked as "to be paid" since the date specified.

Preconditions:

- The date is provided in the message.

Messaging: The external system sends an inbound *request_to_be_paid_invoices-2.0* message. Maintenix returns a *send_to_be_paid_invoices-2.0* outbound message.

Result: Maintenix returns all applicable invoices.

Requesting a particular invoice

Goal: This scenario provides the external system with the details of an invoice.

Summary: The external system sends a message identifying the ID of the invoice. The invoice may have any status. Maintenix returns a message containing the details of the invoice.

Preconditions:

- The invoice ID is provided in the message.

Messaging: The external system sends an inbound *request_to_be_paid_invoices_by_id-2.0* message. Maintenix returns a *send_to_be_paid_invoices-2.0* outbound message.

Result: Maintenix returns the invoice.

Requesting all orders modified since a specified date

Goal: To provide the external system with the details of all issued orders in Maintenix that have been modified since a specified date.

Summary: The external system sends a message identifying a date. Maintenix returns a message containing all issued orders that have been modified since the date specified. Orders may be of any type such as repair orders, purchase order, exchange orders, and so on. Modifications can include changes to the order details, adding part or miscellaneous lines, editing lines, or changing the status of the order. Modifications to orders that have not yet been issued are not returned.

Preconditions:

- The date is provided in the message.

Messaging: The external system sends an inbound *request-order-information-4.0* message. Maintenix returns a *send-order-information-4.0* outbound message.

Result: Maintenix returns all applicable orders.

Requesting a particular order

Goal: To provide the external system with the details of a particular order.

Summary: The external system sends a message identifying the ID of the order. The order may have any status. Maintenix returns a message containing the details of the order.

Preconditions:

- The order ID is provided in the message.

Messaging: The external system sends an inbound *request_order-information-by-id-4.0* message. Maintenix returns a *send-order-information-4.0* outbound message.

Result: Maintenix returns the order information.

Notification service scenarios

The Finance API supports the following notification service scenarios:

- Publishing a detailed financial log automatically.
- Publishing an invoice marked as "to be paid" automatically.
- Publishing order modifcation automatically.
- Publishing an order inventory received and inspected as serviceable automatically.
- Publishing an order inventory returned to vendor automatically.

Publishing a detailed financial log automatically

Goal: To publish a detailed financial log to an external system automatically.

Summary: Maintenix sends a message with a detailed financial log each time a financial transaction is logged in Maintenix.

Preconditions:

- A transaction is logged in Maintenix.

Messaging: Maintenix publishes a *send-detailed-inventory-financial-log-3.0* outbound message.

Result: Maintenix publishes a message for a transaction that is recorded in Maintenix.

Publishing an invoice marked as "to be paid" automatically

Goal: To publish the details of an invoice that has been marked as "to be paid" to an external system automatically.

Summary: Maintenix sends a message providing the details of an invoice that has been marked as "to be paid."

Preconditions:

- An invoice in Maintenix is marked as "to be paid."

Messaging: Maintenix publishes a *send_to_be_paid_invoices-2.0* outbound message.

Result: Maintenix publishes a message for the invoice that is marked as "to be paid."

Publishing order modifications automatically

Goal: To publish the details of a modified order to an external system automatically.

Summary: Maintenix sends a message providing the details of each order that has been modified in Maintenix.

Preconditions:

- An order is created.
- An order is updated. This includes adding lines, adding/removing taxes, adding/removing charges, assigning/unassigning part requests, converting order lines, updating line prices, editing order details, and adding notes.
- Authorization is requested for an order.
- An order is authorized.
- An order is unauthorized.
- An order is issued.
- An order is acknowledged.
- An order is canceled.

- A budget request is made for an order.
- The budget for an order was overridden.
- The approval process for an order is bypassed for AOG.
- A order is rejected.

Messaging: Maintenix publishes a *send-order-information-4.0* outbound message.

Result: Maintenix publishes a message for each order that is modified in Maintenix.

Publishing an order inventory received and inspected as serviceable automatically

Goal: To publish the order inventory, which is received and inspected as serviceable, to an external system automatically.

Summary: Maintenix sends a message providing the details of an order inventory which is received and inspected as serviceable in Maintenix.

Preconditions:

- Inventory received from an inbound shipment of an order.
- When the inventory is inspected as serviceable, the order is not closed.
- Inventory inspected as serviceable in the following cases.
  - Inventory is inspected as serviceable by Maintenix automatically at the receiving shipment time.
  - Inventory is inspected as serviceable by the user manually after receiving the shipment.

Messaging: Maintenix publishes an *order-inventory-received-1.0* outbound message.

Result: Maintenix publishes a message for an order inventory which is received and inspected as serviceable in Maintenix.

Publishing an order inventory returned to vendor automatically

Goal: To publish the order inventory, which is returned to the vendor, to an external system automatically.

Summary: Maintenix sends a message providing the details of an order inventory, which is returned to vendor through outbound shipment (shipment type = RTNVEN).

Preconditions:

- Inventory was received from inbound shipment of an order and inventory inspected as serviceable.
- The order is not closed.
- The outbound shipment (shipment type = RTNVEN) is completed and the ordered inventory is returned to vendor.

- Inventory is inspected as serviceable by Maintenix automatically at the receiving shipment time.
- Inventory is inspected as serviceable by the user manually after receiving the shipment.

Messaging: Maintenix publishes an *order-inventory-returned-1.0* outbound message.

Result: Maintenix publishes a message for an order inventory which is returned to vendor through outbound shipment (shipment type = RTNVEN).

5

Operational service messages

The following chapter provides detailed information for each of the Human Resources 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.

create-purchase-invoice-4.0

The create-purchase-invoice inbound message is sent by external systems to create an invoice in Maintenix. The `create-purchase-invoice-4.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Create-Purchase-Invoice.xml`: This message corresponds to the *Performing invoice matching in external finance systems* scenario.

In this message, an invoice is created with the status of TOBEPAID.

Message contents

An invoice created with the `create-purchase-invoice` message has a status of TOBEPAID.

If the element <`validate-with-order`> is TRUE, and an invoice is created outside of the configurable `INVOICE_TOLERANCE_PCT` range, then invoices will be created with a status of OPEN instead of TOBEPAID .

Mandatory elements

The following table displays the mandatory elements for the `create-purchase-invoice` 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
	Performing invoice matching in external finance system
<invoice-number>	M
<invoice-external-number>	
<contact> <username>	

Element	Scenarios
	Performing invoice matching in external finance system
<vendor> <code>	**Note** <vendor> <code> is mandatory if <po-number>/<po-line-number> is not provided in the message; if <vendor> <code> is not provided but <po-number>/ <po-line-number> is provided in the message, Maintenix will set the vendor of the invoice to match the vendor of the order.
<vendor> <name>	
<date>	M
<terms-conditions>	
<exchange-rate>	
<cash-discount>	
<cash-discount-expiry>	
<invoice-note>	
<validate-with-order>	M
<invoice-line> <line-number>	M

Element	Scenarios Performing invoice matching in external finance system
<invoice-line> <line-description>	**Note** If <po-number>/<po-line-number> are provided in the message and order line is NOT a miscellaneous line, Maintenix will set the line description of the invoice to match the line description of the most recent received shipment line; If <po-number>/<po-line-number> are provided in the message and order line is a miscellaneous line, and <line-description> is not provided, Maintenix will set the line description of the invoice to match the line description of the order line.
<invoice-line> <part-number>	D **Note** If <po-number>/<po-line-number> are provided in the message and order line is NOT a miscellaneous line, Maintenix will set the part number of the invoice to match the part number of the most recent received shipment line.
<invoice-line> <invoice-quantity>	M

Element	Scenarios Performing invoice matching in external finance system
<invoice-line> <invoice-quantity-unit>	D **Note** <invoice-quantity-unit> is mandatory if <po-number>/<po-line-number> is not provided in the message. If <po-number>/<po-line-number> are provided in the message and <invoice-quantity-unit> is not provided, Maintenix will set the invoice quantity unit of the invoice to match the quantity unit of the order line.
<invoice-line> <unit-price>	M
<invoice-line> <tax> <code>	D
<invoice-line> <tax> <amount>	
<invoice-line> <charge> <code>	D
<invoice-line> <charge> <amount>	
<invoice-line> <account> <code>	D **Note** <account-count> is mandatory if <po-number>/<po-line-number> is not provided in the message. If<po-number>/<po-line-number> are provided in the message and <account-code> is not provided, Maintenix will set the account of the invoice to match the account of the order line.
<invoice-line> <account> <type>	D
<invoice-line> <account> <external-reference>	D
<invoice-line> <account> <t-code>	D
<invoice-line> <line-note>	

Element	Scenarios
	Performing invoice matching in external finance system
<invoice-line> <purchase-order> <po-number>	D
<invoice-line> <purchase-order> <po-line-number>	D

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.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<invoice-number>	PO Invoice Details page, PO Invoice Information group box, PO Invoice Number field.
<invoice-external-number>	PO Invoice Details page, Details tab, External Reference field.
<contact> <username>	PO Invoice Details page, Details tab, PO Invoice Contact field.
<vendor> <code>	PO Invoice Details page, PO Invoice Information tab, Vendor field.
<vendor> <name>	PO Invoice Details page, PO Invoice Information tab, Vendor field.
<vendor> <account>	PO Invoice Details page, Details tab, Vendor Account field.
<date>	PO Invoice Details page, Details tab, PO Invoice Date field.
<terms-conditions>	PO Invoice Details page, Details tab, Terms & Conditions field.
<currency>	PO Invoice Details page, Details tab, PO Invoice Total field.

Element	Corresponding entry in Maintenix GUI
<exchange-rate>	PO Invoice Details page, Details tab, Exchange Rate @ PO Invoice field.
<cash-discount>	PO Invoice Details page, Details tab, Cash Discount field.
<cash-discount-expiry>	PO Invoice Details page, Details tab, Discount Expiry Date field.
<invoice-note>	PO Invoice Details page, Details tab, PO Invoice Note group box.
<invoice-line> <line-number>	PO Invoice Details page, PO Invoice Lines tab, Line No column.
<invoice-line> <line-description>	PO Invoice Details page, PO Invoice Lines tab, Line Description column.
<invoice-line> <part-number>	PO Invoice Details page, PO Invoice Lines tab, Line Description column.
<invoice-line> <invoice-quantity>	PO Invoice Details page, PO Invoice Lines tab, Qty column.
<invoice-line> <invoice-quantity-unit>	PO Invoice Details page, PO Invoice Lines tab, Qty column.
<invoice-line> <unit-price>	PO Invoice Details page, PO Invoice Lines tab, Unit Price column.
<invoice-line> <tax> <code>	PO Invoice Details page, PO Invoice Lines tab, Taxes column.
<invoice-line> <tax> <amount>	PO Invoice Details page, PO Invoice Lines tab, Taxes column.
<invoice-line> <charge> <code>	PO Invoice Details page, PO Invoice Lines tab, Charges column.
<invoice-line> <charge> <amount>	PO Invoice Details page, PO Invoice Lines tab, Charges column.
<invoice-line> <account> <code>	PO Invoice Details page, PO Invoice Lines tab, Charge To Account column.
<invoice-line> <account> <type>	Account Details page, Details tab, Account Type column.
<invoice-line> <account> <external-reference>	Account Details page, Details tab, External Reference column.

Element	Corresponding entry in Maintenix GUI
<invoice-line> <account> <t-code>	Account Details page, Details tab, T-Code column.
<invoice-line> <line-note>	PO Invoice Details page, PO Invoice Lines tab, Note column.
<invoice-line> <purchase-order> <po-number>	PO Invoice Details page, PO Invoice Lines tab, Purchase Order / PO Num column.
<invoice-line> <purchase-order><po-line-number>	PO Invoice Details page, PO Invoice Lines tab, Purchase Order / Line No column.

purchase-invoice-created-1.1

The `purchase-invoice-created` outbound message is returned to the external system upon a successful create-purchase-invoice inbound message.

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

mark_invoice_as_paid-2.0

The `mark_invoice_as_paid` inbound message is sent by external systems to update the status of an invoice in Maintenix to PAID.

The `mark_invoice_as_paid-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- Mark_Invoice_as_Paid.xml: This message corresponds to the *Performing invoice matching in external finance systems*, and *Performing invoice matching in Maintenix* scenarios.

In this message, an invoice is identified to be marked as paid.

Message contents

In order to identify the invoice, you must provide either the invoice number or the external reference key in your message. The order must be validated for payment (have a status of TOBEPAID) before you can mark it as paid with the Finance API.

Mandatory elements

The following table displays the mandatory elements for the `mark_invoice_as_paid` 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	
	Performing invoice matching in external finance system	Performing invoice matching in Maintenix
<payment_date>	M	M
<invoice_number>	M* D	M* D
<ext_key_sdesc>	M* D	M* D
<vendor_code>	D	D
<note>		

* You must provide either the <invoice_number> or the <ext_key_sdesc>.

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.
- Invoice does not exist in Maintenix.
- Invoice is not in the status of TOBEPAID.
- Multiple invoice numbers for the same vendor exist that are not closed.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<payment_date>	PO Invoice Details page, Details tab, Closed on Date field.

Element	Corresponding entry in Maintenix GUI
<invoice_number>	PO Invoice Details page, PO Invoice Information group box, PO Invoice Number field.
<ext_key_sdesc>	PO Invoice Details page, Details tab, External Reference field.
<vendor_code>	PO Invoice Details page, PO Invoice Information group box, Vendor field.
<note>	PO Invoice Details page, Details tab, PO Invoice Note group box.

invoice_paid-1.0

The `invoice_paid` outbound message is returned to the external system upon a successful `mark_invoice_as_paid` inbound message.

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

update_currency_exchange_rate-2.0

The `update_currency_exchange_rate` inbound message is sent by external systems to modify the exchange rate.

The `update_currency_exchange_rate-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Updating_Currency_Exchange_Rate.xml`: This message corresponds to the *Updating the currency exchange rate* scenario.

In this message, a currency is identified and its current exchange rate is updated.

Message contents

You can update the exchange rate of currencies already existing in Maintenix. You cannot use the `update_currency_exchange_rate` message to create a new currency in Maintenix. To create a new currency in Maintenix, you must add it to the `REF_CURRENCY` table in the Maintenix database.

Mandatory elements

The following table displays the mandatory elements for the `update_currency_exchange_rate` 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
	Updating the currency exchange rate
<code>	M D
<exchange_rate>	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 currency code and exchange rate are stored in the `REF_CURRENCY` table in the Maintenix database.

currency_exchange_rate_updated-1.0

The `currency_exchange_rate_updated` outbound message is returned to the external system upon a successful `update_currency_exchange_rate` inbound message.

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

create-financial-accounts-request-1.0

The `create-financial-accounts-request` inbound message is sent by external systems to create a financial account in Maintenix.

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

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Creating_a_Financial_Account.xml`: This message corresponds to the *Creating a financial account* scenario. In this message, a new account is created in Maintenix.
- `Creating_a_Financial_Sub-Account.xml`: This message corresponds to the *Creating a financial sub-account* scenario. In this message, a new sub-account is created in Maintenix.
- `Creating_a_Financial_Account_Hierarchy.xml`: This message corresponds to the *Creating a financial account hierarchy* scenario. In this message, a new account hierarchy is created in Maintenix.

Message contents

You can use the `create-financial-accounts-request` message to create a single account or sub-account, or you can create a complete account hierarchy of accounts and their sub-accounts. If a message contains multiple accounts, and one account causes the message to fail, no accounts are created and you must reprocess the entire message.

Mandatory elements

The following table displays the mandatory elements for the create-financial-accounts-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		
	Creating a financial account	Creating a financial sub-account	Creating a financial account hierarchy
<code>	M U	M U	M U
<external-id>	U	U	U
<name>	M	M	M
<description>			
<parent-account-code>		M	
<type>	M D	M D	M D
<t-code>			
<default>			
<subaccount>			M

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.
- The account code exists in Maintenix already.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<code>	Account Details page, Account Code field.
<external-id>	Account Details page, Details tab, External Reference field.
<name>	Account Details page, Account Name field.
<description>	Account Details page, Details tab, Description group box.
<parent-account-code>	Account Details page, Details tab, Parent Account field.

Element	Corresponding entry in Maintenix GUI
<type>	Account Details page, Details tab, Account Type field.
<t-code>	Account Details page, Details tab, T-Code field.
<default>	Account Details page, Details tab, Default check box.
<subaccount>	Account Details page, Details tab, Sub Accounts group box.

create-financial-accounts-response-1.0

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

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

update-financial-accounts-request-1.0

The `update-financial-accounts-request` inbound message is sent by external systems to update a financial account in Maintenix.

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

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Updating_a_Financial_Account.xml`: This message corresponds to the *Updating a financial account* scenario.

In this message, the account and sub-accounts created in the `Creating_a_Financial_Account_Hierarchy.xml` sample message are updated.

Message contents

You can use the `update-financial-accounts-request` message to update the following details for one or more accounts in Maintenix:

- Name
- Description
- Account type

- T-code
- Default account flag

Use the <code> or <external-id> element to identify the account that you want to update. Use all other elements to specify the account updates: each element that is provided in the message, updates the corresponding account information in Maintenix. If an element is not specified in the message, then the corresponding account information is not altered in Maintenix.

You cannot use the `update-financial-accounts-request` message to change the account hierarchy in any way.

Mandatory elements

The following table displays the mandatory elements for the `update-financial-accounts-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
	Updating a financial account
<id> <code>	M D
<id> <external-id>	M **Note** Either the <code> or the <external-id> is mandatory in the message.
<name>	
<description>	
<type>	D
<t-code>	
<default>	

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.
- The account does not exist in Maintenix.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<code>	Account Details page, Account Code field.
<external-id>	Account Details page, Details tab, External Reference field.
<name>	Account Details page, Account Name field.
<description>	Account Details page, Details tab, Description group box.
<type>	Account Details page, Details tab, Account Type field.
<t-code>	Account Details page, Details tab, T-Code field.
<default>	Account Details page, Details tab, Default check box.

update-financial-accounts-response-1.0

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

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

close-financial-accounts-request-1.0

The `close-financial-accounts-request` inbound message is sent by external systems to close a financial account in Maintenix.

The `close-financial-accounts-request-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Closing_a_Financial_Account.xml`: This message corresponds to the *Closing a financial account* scenario.

In this message, the account and sub-accounts created in the `Creating_a_Financial_Account_Hierarchy.xml` sample message are closed.

Message contents

You can use the `close-financial-accounts-request` message to close multiple accounts.

Mandatory elements

The following table displays the mandatory elements for the `close-financial-accounts-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
	Closing a financial account
<account> <code>	M D
<account> <external-id>	M **Note** Either the <code> or the <external-id> is mandatory in the message.

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.
- The account does not exist in Maintenix.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<account> <code>	Account Details page, Account Code field.
<account> <external-id>	Account Details page, Details tab, External Reference field.

close-financial-accounts-response-1.0

The `close-financial-accounts-response` outbound message is returned to the external system upon a successful `close-financial-accounts-request` inbound message.

The `close-financial-accounts-response-1.0.xsd` schema contains annotations describing each element.

open-financial-accounts-request-1.0

The `open-financial-accounts-request` inbound message is sent by external systems to open a financial account in Maintenix.

The `open-financial-accounts-request-1.0.xsd` schema contains annotations describing each element.

Endpoint: http://<maintenix_server>:<http_port>/finance/AccountService

WSDL link: http://<maintenix_server>:<http_port>/finance/AccountService?wsdl

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Opening_a_Financial_Account.xml`: This message corresponds to the *Opening a financial account* scenario.

Message contents

You can use the `open-financial-accounts-request` to open multiple accounts.

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

The following table displays the mandatory elements for the `open-financial-accounts-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
	Opening a financial account
<account> <code>	M D
<account> <external-id>	M D

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.
- The account does not exist in Maintenix.
- More than one accounts are found in the Maintenix for the given external-id.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<account> <code>	Account Details page, Account Code field.
<account> <external-id>	Account Details page, Details tab, External Reference field.

open-financial-accounts-response-1.0

The `open-financial-accounts-response` outbound message is returned to the external system upon a successful `open-financial-accounts-request` inbound message.

The `open-financial-accounts-response-1.0.xsd` schema contains annotations describing each element.

approve_order_budget-2.0

The `approve_order_budget-2.0` inbound message is sent by external systems to approve a budget request.

The `approve_order_budget-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Approving_Budget_Order_Check.xml`: This message corresponds to the *Updating the currency exchange rate* scenario.

In this message, a budget request is approved.

Message contents

The message contains the number of the order to approve, the optional budget check reference code, and an optional note.

Mandatory elements

The following table displays the mandatory elements for the `approve_order_budget` 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 Approving a budget order check
<order_number>	M D
<budget-check-reference>	
<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.
- The currency does not exist in Maintenix.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<order_number>	PO Invoice Details page, Order Number field.
<budget-check-reference>	PO Invoice Details page, Budget Check Reference field.
<note>	PO Invoice Details page, Authorization tab, Note column.

reject_order_budget-2.0

The `reject_order_budget` inbound message is sent by external systems to reject a budget request.

The `reject_order_budget-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Rejecting_Budget_Order_Check.xml`: This message corresponds to the *Rejecting a budget order check* scenario.

In this message, a budget request is rejected.

Message contents

The message contains he number of the order to approve, the optional budget check reference code, and an optional note.

Mandatory elements

The following table displays the mandatory elements for the `reject_order_budget` 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
	Rejecting a budget order check
<order_number>	M D
<budget-reference-check>	
<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.
- The currency does not exist in Maintenix.

Mapping elements to Maintenix

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

Element	Corresponding entry in Maintenix GUI
<order_number>	PO Invoice Details page, Order Number field.
<budget-check-reference>	PO Invoice Details page, Budget Check Reference field.
<note>	PO Invoice Details page, Authorization tab, Note column.

6

Querying service messages

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

For each message, the following information is provided:

- A description of the message processing.
- A description of each of the sample messages that complete the supported scenarios.
- For the response messages, a table with the corresponding fields in Maintenix for each element.

request-detailed-inventory-financial-log-3.0

The `request-detailed-inventory-financial-log` inbound message is sent by the external system. Maintenix returns the requested information in a `send-detailed-inventory-financial-log` outbound message.

The `request-detailed-inventory-financial-log-3.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Detailed_Financial_Log_for_Financial_Period.xml`: This message corresponds to the *Requesting a detailed financial log for a financial period* scenario.

In this message, the start and end dates for the financial log request is sent.

Message contents

Maintenix logs financial transactions related to inventory. The `request-detailed-inventory-financial-log` message retrieves the logged transactions for the time period specified.

request-detailed-inventory-financial-log-by-id-3.0

The `request-detailed-inventory-financial-log-by-id` inbound message is sent by the external system. Maintenix returns the requested information in a `send-detailed-inventory-financial-log` outbound message.

The `request-detailed-inventory-financial-log-by-id-3.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Detailed_Financial_Log_for_Transaction.xml`: This message corresponds to the *Requesting a detailed financial log for a transaction* scenario.

In this message, the transaction ID for the financial log request is sent.

Message contents

The **transaction-id** element must contain the `XACTION_DB_ID` followed by the `XACTION_ID`. For example:

```
<transaction-id>4650:100008</transaction-id>
```

request_summary_inventory_financial_log-2.0

The `request_summary_inventory_financial_log` inbound message is sent by the external system. Maintenix returns the requested information in a `send_summary_inventory_financial_log` outbound message.

The `request_summary_inventory_financial_log-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Summary_of_Account_Transactions.xml`: This message corresponds to the *Requesting a summary of account transactions for a financial period* scenario.

In this message, the start and end dates for the financial log request are sent.

Message contents

The `request_summary_inventory_financial_log` message returns a full list of accounts affected by inventory-related financial transactions for the time period specified. The credits and debits occurring within the specified time period for each account are calculated, and the total debit or credit is returned to the external system. Any credits or debits for an affected account that occur outside of the time period specified in the message are not calculated into the total credit or debit.

send_summary_inventory_financial_log-2.0

Maintenix sends a `request_summary_inventory_financial_log` outbound message following a `send_summary_inventory_financial_log` inbound message from the external system.

The `send_summary_inventory_financial_log-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Sending_Summary_Inventory_Financial_Log.xml`: This message corresponds to the *Requesting a summary of account transactions for a financial period* scenario.

In this message, the information requested in `Requesting_Summary_of_Account_Transactions.xml` is returned to the external system.

Message contents

The `send_summary_inventory_financial_log` message contains the calculated total of the debit and credit transactions for each account within the specified financial period. If the total amount is a credit, then the <total_credit_amount> element is populated with the value. If the total amount is a debit, the <total_debit_amount> element is populated with the value.

Mapping elements to the Maintenix GUI

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

Element	Corresponding entry in Maintenix GUI
<account> <code>	Transaction Details page, Accounts Affected tab, Account column.
<account> <type>	Transaction Details page, Accounts Affected tab, Account Type column.
<account> <external_reference>	Transaction Details page, Accounts Affected tab, External Reference column.
<t_code>	Transaction Details page, Accounts Affected tab, T-Code column.
<total_credit_amount>	N/A.
<total_debit_amount>	N/A.

request_to_be_paid_invoices-2.0

The `request_to_be_paid_invoices` inbound message is sent by the external system. Maintenix returns the requested information in a `send_to_be_paid_invoices` outbound message.

The `request_to_be_paid_invoices-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Invoices_Marked_To_Be_Paid_Since_Date.xml`: This message corresponds to the *Requesting all invoices marked "to be paid" since a specified date* scenario.

In this message, the date is sent and all invoices marked TOBOPAID are requested.

Message contents

Only invoices that are in the status of TOBEPAID are returned to the external system.

In Maintenix, the date that an invoice has been marked as *To Be Paid* appears in the Closed on Date field of the PO Invoice Details page.

request_to_be_paid_invoices_by_id-2.0

The `aircraft-status-request` inbound message is sent by the external system. Maintenix returns the requested information in a `send_to_be_paid_invoices-2.0` (p.61) outbound message.

The `request_to_be_paid_invoices_by_id-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Particular_Invoice.xml`: This message corresponds to the *Requesting a particular invoice* scenario.

In this message, the invoice ID is sent.

Message contents

The <invoice_id> element must contain the `PO_INVOICE_DB_ID`, followed by the `PO_INVOICE_ID`. For example:

```
<invoice_id>4650:113486</invoice_id>
```

request-order-information-4.0

The `request-order-information` inbound message is sent by the external system. Maintenix returns the requested information in a `send_order_information` outbound message.

The `request-order-information-4.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Orders_Modified_Since_Date.xml`: This message corresponds to the *Requesting all orders modified since a specified date* scenario.

In this message, the date is sent, since which all issued orders that have been modified are requested.

Message contents

The `request-order-information` message returns only orders that have a status of ISSUED. After an order has been issued, the request-order-information message will return all orders that have been modified in any of the following ways:

- Order details are added, updated, or removed.
- Part lines are added, updated, or removed.
- Miscellaneous lines are added, updated, or removed.
- The order is canceled.

Other order modifications such as adding technical references, adding part requests, or changing the authorization are not picked up by the `request-order-information` message.

In Maintenix, the date that an order has been modified corresponds to the Last Edit Date field on the Order Details page, Details tab. This field is only visible if the order has been issued.

request_order-information-by-id-4.0

The `request-order-information-by-id` inbound message is sent by the external system. Maintenix returns the requested information in a `send_order_information` outbound message.

The `request-order-information-by-id-4.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Requesting_Particular_Order.xml`: This message corresponds to the *Requesting a particular order* scenario.

In this message, the order ID is sent.

Message contents

The <order_id> element must contain the `PO_DB_ID` followed by the `PO_ID`. For example:

```
<order_id>4650:107043</order_id>
```

7

Notification Service Messages

The following section provides detailed information for each of the Finance 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.

Refer *Configuring Maintenix for the external system's web service* for more information on configuring the Maintenix notification service.

send-detailed-inventory-financial-log-3.0

The `send-detailed-inventory-financial-log` outbound message is sent to the external system as a notification service message, or as a response to either the *request-detailed-inventory-financial-log-3.0* or the *request-detailed-inventory-financial-log-by-id-3.0* inbound querying service messages from an external system.

To enable the notification service message, you must provide the following `NAMESPACE` and the `ROOT_NAME` in Maintenix's `INT_DEST_LOOKUP` table. For more information, see *Configuring Maintenix for the external system's web service*.

- ***NAMESPACE***: *http://xml.mxi.com/xsd/core/finance/send-detailed-inventory-financial-log/3.0*
- ***ROOT_NAME***: *send-detailed-inventory-financial-log*

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

TRIGGER_ID	TRIGGER_CD	TRIGGER_NAME
99912	MX_FNC_SEND_DETAILED_FINANCIAL_LOG	Send Detailed Financial Log

The `send-detailed-inventory-financial-log-3.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- Sending_Detailed_Inventory_Financial_Log.xml: This message corresponds to the *request-detailed-inventory-financial-log-by-id-3.0* scenario. In this message, the information requested in Requesting_Detailed_Inventory_Financial_Log_by_ID.xml is returned to the external system.
- Publishing_Detailed_Inventory_Financial_Log.xml: This message corresponds to the *Publishing a detailed financial log automatically* scenario. In this message, the financial transaction log is published for an invoice that has been marked as "to be paid."

Message triggers

The `send-detailed-inventory-financial-log` notification message is published to external systems when a transaction is recorded in Maintenix. Transactions are logged for the following events:

- Inventory is created.
- Inventory is received.
- Inventory is returned to a vendor.
- An invoice is validated for payment.
- Inventory is issued.
- An inventory issue is undone.
- Inventory is turned in.
- The owner of the inventory is changed.
- The quantity of consumable parts is adjusted.
- The unit price of a consumable part is adjusted.
- Consumable inventory is scrapped.
- Consumable inventory is archived.
- Consumable inventory is unarchived.

Mapping elements to the Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `send-detailed-inventory-financial-log` message elements.

Element	Corresponding entry in Maintenix GUI
<id>	Transaction Details page, page title.
<type>	Transaction Details page, Details group box, Transaction Type field.
<date>	Transaction Details page, Details group box, Transaction Date field.
<description>	Transaction Details page, Details group box, Description field.
<part-number-oem>	Part Details page, OEM Part Number field.
<part-number-description>	Part Details page, Name field.
<manufacturer-code>	Part Details page, Manufacturer field.
<aircraft-registration-code>	Inventory Details page, Details tab, Aircraft Information band, Registration Code field.
<transaction-account> <code>	Transaction Details page, Accounts Affected tab, Account column.
<transaction-account> <type>	Transaction Details page, Accounts Affected tab, Account Type column.
<transaction-account> <external-reference>	Transaction Details page, Accounts Affected tab, External Reference column.
<transaction-account> <t-code>	Transaction Details page, Accounts Affected tab, T-Code column.
<transaction-account> <credit>	Transaction Details page, Accounts Affected tab, Credit column.
<transaction-account> <debit>	Transaction Details page, Accounts Affected tab, Debit column.

send-order-information-4.0

The `send-order-information` outbound message is sent to the external system as a notification service message, or as a response to either the *send-order-information-4.0* or the *request_order-information-by-id-4.0* inbound querying service messages from an external system.

To enable the notification service message, you must provide the following NAMESPACE and the ROOT_NAME in Maintenix's INT_DEST_LOOKUP table. For more information, see *Configuring Maintenix for the external system's web service*.

- ***NAMESPACE:*** *http://xml.mxi.com/xsd/core/finance/send-order-information/4.0*
- ***ROOT_NAME:*** *send-order-information*

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

TRIGGER_ID	TRIGGER_CD	TRIGGER_NAME
99913	MX_FNC_SEND_ORDER_INFORMATION	Send Order Information

The send-order-information-4.0.xsd schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- Sending_Order_Information.xml: This message corresponds to the *Requesting all orders modified since a specified date* scenario. In this message, the information requested in Requesting_Orders_Modified_Since_Date.xml is returned to the external system.
- Publishing_Order_Information.xml: This message corresponds to the *Publishing order modifications automatically*, *Performing invoice matching in external finance systems*, and *Performing invoice matching in Maintenix* scenarios. In this message, the details of an order that has been modified to include an additional part line is published to the external system.

Message triggers

The send-order-information notification message is published to external systems when one of the following events occurs in Maintenix:

- An order is created.
- An order is updated. This includes adding lines, adding/removing taxes, adding/removing charges, assigning/unassigning part requests, converting order lines, updating line prices, editing order details, and adding notes.
- Authorization is requested for an order.
- An order is authorized.
- An order is unauthorized.
- An order is issued.
- An order is acknowledged.

- An order is canceled.
- A budget request is made for an order.
- The budget for an order was overridden.
- The approval process for an order is bypassed for AOG.
- A order is rejected.

The <action> element of the `send-order-information` outbound message identifies the event that triggered the message. When the message is sent as a response to a query from an external system, the contents of the <action> element is `response`. The following table identifies all possible values for the <action> element.

Value	Description
create	An order is created.
update	An order is modified (by an action not otherwise listed).
auth	A user authorizes an order (not including a AOG bypass approval).
unauth	A user reverses their authorization.
issue	An order issued.
ack	An order is acknowledged.
cancel	An order is canceled.
response	A request was made for the details of an order.
addline	A line was added to the order.
budgreq	A budget request is made for an order.
budgover	The budget for an order was overridden.
bypass	The approval process for an order is bypassed for AOG.
reject	A user rejects an order.

Mapping elements to the Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `send-order-information` message elements.

Element	Corresponding entry in Maintenix GUI
<order-number>	Order Details page, Order Information group box, Order Number field.

Element	Corresponding entry in Maintenix GUI
<order-type>	Order Details page, Order Information group box, Type field.
<cancelled>	
<issued>	Order Details page, Order Information group box, Status field.
<authorized-status>	Order Details page, Order Information group box, Authorization Status.
<budget-check-reference>	
<budget-check-code>	
<contact>	Order Details page, Details tab, Purchasing Contact field.
<vendor> <code>	Order Details page, Details tab, Vendor field.
<vendor> <name>	Order Details page, Details tab, Vendor field.
<vendor> <account>	Order Details page, Details tab, Vendor Account field.
<vendor> <external-key>	Vendor Details page, Details tab, External Key field.
<broker>	
<priority>	Order Details page, Details tab, Priority field.
<bill-to>	Order Details page, Details tab, Bill To field.
<consign-to>	Order Details page, Details tab, Consign To field.
<re_ship_to>	
<ship-to-location>	Order Details page, Details tab, Ship To field.
<transportation-type>	Order Details page, Details tab, Transportation Type field.
<currency>	Order Details page, Details tab, Total Price field.
<exchange-rate>	Order Details page, Details tab, Exchange Rate @ Issue field.
<terms-conditions>	Order Details page, Details tab, Terms & Conditions field.
<fob>	Order Details page, Details tab, Freight On Board field.
<order-external-reference>	Order Details page, Details tab, External Reference field.
<customer-code>	Order Details page, Details tab, Customer Code field.

Element	Corresponding entry in Maintenix GUI
<ship-to-code>	Order Details page, Details tab, Ship To Code field.
<note-to-vendor>	Order Details page, Details tab, Note To Vendor group box.
<borrow-rate>	Order Details page, Details tab, Borrow Rate field.
<order-line>	Order Details page, Order Lines tab.
<order-creation-date>	Order Details page, History tab, Date field.

send_to_be_paid_invoices-2.0

The `send_to_be_paid_invoices` outbound message is sent to the external system as a notification service message, or as a response to either the *request_to_be_paid_invoices-2.0* or the *request_to_be_paid_invoices_by_id-2.0*inbound querying service messages from an external system.

To enable the notification service message, you must provide the following `NAMESPACE` and the `ROOT_NAME` in Maintenix's `INT_DEST_LOOKUP` table. For more information, see *Configuring Maintenix for the external system's web service*.

- ***NAMESPACE:*** *http://xml.mxi.com/xsd/core/finance/send_to_be_paid_invoices/2.0*
- ***ROOT_NAME:*** *send_to_be_paid_invoices*

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

TRIGGER_ID	TRIGGER_CD	TRIGGER_NAME
99981	MX_FNC_SEND_TO_BE_PAID_INVOICE	Send Detailed To Be Paid Invoice

The `send_to_be_paid_invoices-2.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Sending_To_Be_Paid_Invoices.xml`: This message corresponds to the *Requesting all invoices marked "to be paid" since a specified date* scenario. In this message, the information requested in `Requesting_Invoices_Marked_To_Be_Paid_Since_Date.xml` is returned to the external system.

- `Publishing_To_Be_Paid_Invoices.xml`: This message corresponds to the *Publishing an invoice marked as "to be paid" automatically* scenario. In this message, the details of an invoice that has been marked as to be paid is published to the external system.

Message triggers

The `send_to_be_paid_invoices` notification message is published to external systems when the status of an invoice in Maintenix is marked as TOBEPAID.

Mapping elements to the Maintenix GUI

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

Element	Corresponding entry in Maintenix GUI
<invoice_number>	PO Invoice Details page, PO Invoice Information group box, PO Invoice Number field.
<contact> <username>	User Details page, Details group box, Username field.
<contact> <hr_code>	User Details page, Details group box, HR Code field.
<contact> <name>	PO Invoice Details page, Details tab, PO Invoice Contact field.
<vendor> <code>	Vendor Details page, Details group box, Vendor Code field.
<vendor> <name>	PO Invoice Details page, PO Invoice Information group box, Vendor field.
<vendor> <account>	PO Invoice Details page, Details tab, Vendor Account field.
<date>	PO Invoice Details page, Details tab, PO Invoice Date field.
<terms_conditions>	PO Invoice Details page, Details tab, Terms & Conditions field.
<currency>	PO Invoice Details page, Details tab, PO Invoice Total field.
<exchange_rate>	PO Invoice Details page, Details tab, Exchange Rate @ PO Invoice field.
<cash_discount>	PO Invoice Details page, Details tab, Cash Discount field.
<invoice_line> <line_number>	PO Invoice Details page, PO Invoice Lines tab, Line No column.
<invoice_line> <line_description>	PO Invoice Details page, PO Invoice Lines tab, Line Description column.

Element	Corresponding entry in Maintenix GUI
<invoice_line> <part_number>	PO Invoice Details page, PO Invoice Lines tab, Line Description column.
<invoice_line> <invoice_quantity>	PO Invoice Details page, PO Invoice Lines tab, Qty column.
<invoice_line> <invoice_quantity_unit>	PO Invoice Details page, PO Invoice Lines tab, Qty column.
<invoice_line> <unit_price>	PO Invoice Details page, PO Invoice Lines tab, Unit Price column.
<invoice_line> <tax>	PO Invoice Details page, PO Invoice Lines tab, Taxes column.
<invoice_line> <account> <code>	PO Invoice Details page, PO Invoice Lines tab, Charge To Account column.
<invoice_line> <account> <type>	Account Details page, Details tab, Account Type field.
<invoice_line> <account> <external_reference>	Account Details page, Details tab, External Reference field.
<invoice_line> <account> <t_code>	Account Details page, Details tab, T-Code field.
<invoice_line> <line_note>	PO Invoice Details page, PO Invoice Lines tab, Note column.
<invoice_line> <purchase_order> <po_number>	PO Invoice Details page, PO Invoice Lines tab, Purchase Order - PO Num column.
<invoice_line> <purchase_order> <po_line_number>	PO Invoice Details page, PO Invoice Lines tab, Purchase Order - Line No column.

order-inventory-received-1.0

The `order-inventory-received` outbound message is sent to the external system as a notification service message.

To enable the notification service message, you must provide the following `NAMESPACE` and the `ROOT_NAME` in Maintenix's `INT_DEST_LOOKUP` table. For more information, see *Configuring Maintenix for the external system's web service*.

- ***NAMESPACE:*** *http://xml.mxi.com/xsd/core/finance/order-inventory-received/1.0*
- ***ROOT_NAME:*** *order-inventory-received*

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

TRIGGER_CD	TRIGGER_NAME
MX_IN_INSPSRV	Send Order Inventory Received

The `order-inventory-received-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Publishing_order-inventory_received.xml`: This message corresponds to Publishing an order inventory which is received and inspected as serviceable automatically *Publishing an order inventory received and inspected as serviceable automatically* scenario. In this message, the details of an order inventory, which is received and inspected as serviceable, is published to the external system.

Message triggers

The `order-inventory-received` notification message is published to external systems when one of the following events occurs in Maintenix:

- Inventory received from an inbound shipment of an order. Part does NOT require inspection. When received it will be marked as serviceable automatically by the system.
- Inventory received from an inbound shipment of an order. Part requires inspection. When received it will be marked as serviceable manually by the user.
- A kit received from an inbound shipment of an order is ready for issue (RFI) after all the kit's inventory items are RFI (either when received or after inspection as serviceable). A single message is sent for the kit. (Messages are not sent for individual items in the kit.

Mapping elements to the Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `order-inventory-received` message elements.

Element	Corresponding entry in Maintenix GUI
<current-date>	N/A. It is the system date when the message is generated .

Element	Corresponding entry in Maintenix GUI
<part-number>	Order Details page, Receipt & Returns tab, Inventory Receipts section, Line Description field. E.g. If Line Description Field = CF6-80A2 (General Electric CF6-80A2), this element will be populated as CF6-80A2.
<manufacturer-code>	Order Details page, Receipt & Returns tab, Inventory Receipts section, click on link of Line Description field, Part Details Page, Manufacturer field. E.g. If Manufacturer field= 2000 (General Electric), this element will be populated as 2000.
<standard-quantity>	Order Details page, Receipt & Returns tab, Inventory Receipts section, Qty field. E.g. If Qty Field = 1 EA, this element will be populated as 1.
<standard-unit>	Order Details page, Receipt & Returns tab, Inventory Receipts section, Qty field. For example, if Qty Field = 1 EA, this element will be populated as EA.
<vendor-quantity>	N/A. It is calculated value based on the standard quantity and conversion of standard unit and vendor unit.
<vendor-unit>	Order Details page, Receipt & Returns tab, Inventory Receipts section, Ordered Qty field. E.g. If Ordered Qty Field = 1 BOX, this element will be populated as BOX.
<order-number>	Order Details page, Order Information group box, Order Number field.
<order-line-number>	Order Details page, Receipt & Returns tab, Inventory Receipts section, Line No field.
<order-creation-date>	Order Details page, History tab, Date field.

order-inventory-returned-1.0

The `order-inventory-returned` outbound message is sent to the external system as a notification service message.

To enable the notification service message, you must provide the following `NAMESPACE` and the `ROOT_NAME` in Maintenix's `INT_DEST_LOOKUP` table. For more information, see *Configuring Maintenix for the external system's web service*.

- ***NAMESPACE :*** *http://xml.mxi.com/xsd/core/finance/order-inventory-returned/1.0*

- ***ROOT_NAME:*** *order-inventory-returned*

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

TRIGGER_CD	TRIGGER_NAME
MX_IX_SEND	Send Order Inventory Returned

The `order-inventory-returned-1.0.xsd` schema contains annotations describing each element.

Sample messages

The following sample messages provide examples of the supported scenarios:

- `Publishing_order-inventory_returned.xml`: This message corresponds to the *Publishing an order inventory returned to vendor automatically* scenario. In this message, the details of an order inventory, which is returned to the vendor, is published to the external system.

Message triggers

The `order-inventory-returned` notification message is published to external systems when one of the following events occurs in Maintenix:

- The outbound shipment (shipment type = RTNVEN) is completed and the ordered inventory is returned to the vendor.
  - Inventory was received from inbound shipment of an order and inventory inspected as serviceable.
  - The order is not closed.

Mapping elements to the Maintenix GUI

The following table displays the corresponding field in the Maintenix user interface for each of the `order-inventory-returned` message elements.

Element	Corresponding entry in Maintenix GUI
<current-date>	N/A. It is the system date when the message is generated.
<part-number>	Order Details page, Receipt & Returns tab, Outbound Shipment section, click the link of the shipment field (the shipment type = RTNVEN), Shipment Detail Page, Shipment Lines Section, OEM Part No field.

Element	Corresponding entry in Maintenix GUI
<manufacturer-code>	Order Details page, Receipt & Returns tab, Outbound Shipment section, click on link of the shipment field (the shipment type = RTNVEN), Shipment Detail Page, Shipment Lines Section, click the OEM Part No field, Part Details Page, Manufacturer field. E.g. If Manufacturer field= 2000 (General Electric), this element will be populated as 2000.
<standard-quantity>	Order Details page, Receipt & Returns tab, Outbound Shipment section, click on link of the shipment field (the shipment type = RTNVEN), Shipment Detail Page, Shipment Lines Section, Received QTY field. E.g. If Received QTY field = 6 EA, this element will be populated as 6.
<standard-unit>	Order Details page, Receipt & Returns tab, Outbound Shipment section, click the link of the shipment field (the shipment type = RTNVEN), Shipment Detail Page, Shipment Lines Section, Received QTY field. E.g. If Received QTY field = 6 EA, this element will be populated as EA.
<vendor-quantity>	N/A. It is calculated value based on the standard quantity and conversion of the standard unit and vendor unit.
<vendor-unit>	Order Details page, Receipt & Returns tab, Outbound Shipment section, click on the link of the shipment field (the shipment type = RTNVEN), Shipment Detail Page, Shipment Line Section, Expected QTY field. E.g. If Expected QTY field = 6 EA (1 BOX), this element will be populated as BOX .
<order-number>	Order Details page, Order Information group box, Order Number field.
<order-line-number>	Order Details page, Receipt & Returns tab, Inventory Receipts section, Line No field .
<order-creation-date>	Order Details page, History tab, Date field.

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.