Oracle® Cloud

Using B2B for Oracle Integration Generation 2

F19388-14
December 2022

Oracle Cloud Using B2B for Oracle Integration Generation 2,

F19388-14

Primary Author: Oracle Corporation

Contents

Preface

This guide describes how to use B2B for Oracle Integration.

Note:

The use of this adapter may differ depending on the features you have, or whether your instance was provisioned using Standard or Enterprise edition. These differences are noted throughout this guide.

Topics:

Audience

This guide is intended for developers who want to use B2B for Oracle Integration.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `https://www.oracle.com/corporate/accessibility/`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `https://support.oracle.com/portal/` or visit `Oracle Accessibility Learning and Support` if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

See these Oracle resources:

- Oracle Cloud

  `http://cloud.oracle.com`
- *Using Integrations in Oracle Integration Generation 2*
- *Using the Oracle Mapper with Oracle Integration Generation 2*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction to B2B for Oracle Integration

Learn about how Oracle Integration provides e-commerce support with B2B for Oracle Integration.

Topics:

Business-to-Business E-Commerce

In B2B e-commerce, an enterprise extends its business processes to reach trading partners. B2B e-commerce represents classic business processes, mature business documents, and industry-tempered messaging services. It requires a unified business process platform, end-to-end instance tracking, visibility and auditing, integrated process intelligence, process and service governance, and centralized security.

About Electronic Data Interchange

Electronic Data Interchange (EDI) covers a set of data formats used to exchange formal business documents between companies.

These data formats have a long history of usage in the industry starting from the 1980s. EDI formats are standardized by various governing bodies. The two major formats are:

- EDI X12 format, governed by the ANSI X12 standards body. EDI X12 is a data format used to exchange specific data between trading partners. The term trading partner may represent an organization, group of organizations, or some other entity. It is typically just an organization or company.
- UN/EDIFACT format, governed by the United Nations Economic Commission for Europe (UNECE).

The governing bodies define formats for several types of business documents, such as a purchase order, invoice, insurance claim, and so on that are used across many vertical industries, such as supply chain, finance, transportation, and insurance. While EDI has a dominant place in B2B (business to business) communications, other formats based on XML are also in use today. B2B for Oracle Integration currently supports the EDI X12 and UN/EDIFACT formats.

Here's an example purchase order in the ANSI EDI X12 format.

```
ISA|00|          |00|          |01|111111T        |01|22222          |
190312|0845|U|00401|000001894|0|T|~
GS|PO|0124578|ACME|20190312|084515|0123456|X|004010
ST|850|1234
BEG|00|NE|45030000000||20190312
CUR|2L|USD
REF|IT|999|Global Chips
REF|WO|P8923.5
REF|KY||Standard Terms and conditions will apply
FOB|PC|OR|ORIGIN FREIGHT COLLECT|02|FOB
ITD||3|||||10|||||P03R
DTM|002|20190329
TD5|B||05||Ground
N1|BT|Global Chips
N2|ACCOUNTS PAYABLE
N3|P.O. BOX 1111
N4|NEW YORK|NY|10001|US
PER|BD|JAMES SMITH|TE|1112223333|EM|jamess@globalchips.com
PO1|00001|1|EA|74.99|PE|BP|5566|VN|AB-1264
PID|F||||AB-1264 BRACKET ASSY WITH SPRING
PO1|00002|1|EA|25.00|PE|BP|7264|VN|DE-1834
PID|F||||DE-1834 GEAR BOX PACKAGE
CTT|2|2
AMT|TT|99.99
SE|22|1234
GE|1|0123456
IEA|1|000001894
```

Here's an example purchase order in the ANSI EDI UN/EDIFACT format.

```
UNB+UNOA:1+US::US+50138::THEM+140531:0305+001934++ORDERS'
UNH+1+ORDERS:91:2:UN'
BGM+220+A761902+4:20140530:102+9'
RFF+CT:EUA01349'
RFF+AAV::C'
TXT+THIS IS WHAT AN EDI MESSAGE WOULD LOOK LIKE... '
NAD+BY++OUR NAME PLC::::+++++EW4 34J'
CTA+PD'
COM+01752 253939:TE+01752 253939:FX+0:TL'
CTA+OC+:A.SURNAME'
COM+2407:EX'
CTA+TI+:B.BROWN'
COM+0:EX'
CTA+SU'
COM+0161 4297476:TE+01752 670633:FX'
UNT+15+1'
UNZ+1+001934'
```

What Is B2B for Oracle Integration

Oracle Integration provides support for B2B e-commerce with B2B for Oracle Integration.

B2B for Oracle Integration represents a collective set of features inside Oracle Integration to support processing, including:

- A trading partner management interface.
- An EDI schema editor for customization.
- A specialized B2B tracking user interface for viewing messages exchanged with trading partners.
- Adapters to support B2B protocols such as AS2 and FTP. See Transport Protocols Supported in B2B for Oracle Integration.
- Actions you place within your integrations for EDI processing. See Business Protocols Supported in B2B for Oracle Integration.

Note:

Some features mentioned above are not available in Oracle Integration. Upgrade to Oracle Integration Generation 2 to receive all B2B for Oracle Integration features.

Business Protocols Supported in B2B for Oracle Integration

B2B for Oracle Integration supports the EDI X12 and EDIFACT business protocols for the exchange of business documents between Oracle Integration and a trading partner. EDI X12 versions 4010 to 8010 and EDIFACT versions D96A (1996) through D20A (2020), which include all document types within each version, are provided with B2B for Oracle Integration.

Transport Protocols Supported in B2B for Oracle Integration

The transport protocol defines the underlying transport used to send and receive data between trading partners. B2B for Oracle Integration supports the following protocols:

- AS2 (Applicability Statement 2)
- FTP and SFTP

The following protocols are available for use in B2B for Oracle Integration, but do not support trading partner mode.

- AQ
- File
- REST
- SOAP
- JMS

Access B2B for Oracle Integration

B2B for Oracle Integration is automatically included when provisioning the following Oracle Integration versions.

Note:

While B2B for Oracle Integration is visible in the left navigation pane for Oracle Integration Standard Edition instances, you are not licensed to use this component unless you update your instance to Oracle Integration Enterprise Edition, which provides complete licensing access to all components. See Editing the Edition, License Type, Message Packs, and Custom Endpoint of an Instance in *Provisioning and Administering Oracle Integration Generation 2*.

Version	Enterprise Edition	Standard Edition
Oracle Integration	Yes	No
Oracle Integration for Oracle SaaS	Yes	No
Oracle Integration Generation 2	Yes	No
Oracle Integration for Oracle SaaS Generation 2	Yes	No

Two Patterns For Using B2B for Oracle Integration

B2B for Oracle Integration can be used in two different ways. This section describes these two patterns for processing B2B messages.

- Standalone mode:
  Uses the basic building blocks such as the AS2, FTP, SOAP, or REST Adapters along with the B2B action to process B2B messages. This is the same as any other integration. You build the B2B integrations yourself and track the processing of B2B messages as integration instances on the Track Instances page. In this mode, the B2B trading partner definitions are not used.
- B2B trading partner mode:
  Extends standalone mode and provides a fully declarative and configuration-driven setup with trading partner management and a specialized B2B message tracking user interface (the Track B2B Messages page). You define trading partner profiles at design-time and use them during runtime. You only create integrations to interface with your backend applications. Behind the scenes, however, message processing still occurs through integrations. The integrations are automatically created for you using templates available with B2B for Oracle Integration. In short, this mode provides you with a ready-to-use solution that you simply configure and use.

The differences between these two modes are as follows. Overall, the B2B trading partner mode is the preferred pattern to use because it is fully-declarative and simpler to use. This mode was introduced recently when compared to the standalone mode.

Note:

The B2B trading partner mode is only available in Oracle Integration Generation 2 and later.

Feature	Standalone Mode	B2B Trading Partner Mode
Processing basic B2B messages, including EDI processing	Available. You design and build the B2B integrations using any technology adapters and actions available in Oracle Integration. For instance, you use the AS2, FTP, REST, or SOAP Adapter to interface with your trading partner. To integrate with backend (or back-office) applications, you also use any available application adapters, such as Oracle ERP Cloud Adapter or Oracle NetSuite Adapter.	Available. You start by defining trading partners and other objects in a specialized B2B user interface. The B2B integrations are automatically created for you using a transport type you choose (for example, AS2 or FTP). You separately create the backend (or back-office) integrations using any available application adapters and link to them inside the trading partner profile.
Tracking B2B message activity	Limited capability. B2B message traffic is tracked solely as integration instances. Each incoming or outgoing message is one integration instance. You use the Track Instances page to view B2B messages. Tracking and troubleshooting requires deeper skills, likely requiring knowledge about how the integrations were built.	Specialized tracking is available. In this mode, B2B messages are persisted separately and a specialized B2B tracking user interface (the Track B2B Messages page) is provided to view them. This page is less technical and designed for use by B2B operations or help desk staff for routine tracking and troubleshooting.
On-boarding new trading partners	To on-board a new trading partner, you manually create and activate additional integrations to receive and send messages from the new trading partner. Design your integrations to support document-based routing. This mode does not use B2B trading partner definitions.	To on-board a new trading partner, you add a trading partner definition in the user interface and input additional configuration. This automatically creates additional integrations to receive and send messages from the new partner. Document-based routing is defined as part of trading partner definitions.
Using any adapter to interface with a trading partner	Because the standalone mode only provides basic build blocks, it gives you the power to design the B2B integration any way you want using any technology adapter available in Oracle Integration. For example, you can use the AS2, FTP, REST, or SOAP Adapter to communicate with a trading partner even though some adapters do not currently provide a B2B trading partner mode.	At this time, only the AS2 and FTP Adapters support the B2B trading partner mode. For example, you cannot currently use the REST or SOAP protocols in this mode.

If you initially built integrations in standalone mode, you can continue to use them as-is. You cannot mix and match the two modes; they work separately and their input/output schemas are different.

More details about each pattern are provided in subsequent chapters.

2

Use B2B for Oracle Integration in Trading Partner Mode

This chapter describes the concepts and tasks for using B2B for Oracle Integration in trading partner mode.

Topics:

Note:

B2B trading partner mode is only available in Oracle Integration Generation 2. If you are using Oracle Integration, you must use standalone mode. See Use B2B for Oracle Integration in Standalone Mode.

Trading Partner Mode Concepts

This section describes trading partner mode concepts in B2B for Oracle Integration.

Topics:

Host Company

There are two parties involved in any document exchange in B2B for Oracle Integration: your company (the host company) and an external trading partner.

In Oracle Integration, selecting **B2B** > **Host Profile** from the left navigation menu invokes the configuration page you must complete on behalf of your company.

You only need to enter B2B identifiers for the host profile. You enter your company's identity information such as the EDI Interchange ID and the AS2 Identifier. Other types of configuration, such as signing certificates for your company, are entered later.

When you send electronic documents to an external trading partner, some of the B2B identifiers you enter in the host profile are inserted into the message. This enables the recipient party to identify your company as the sender of the message.

See Define the Host Profile.

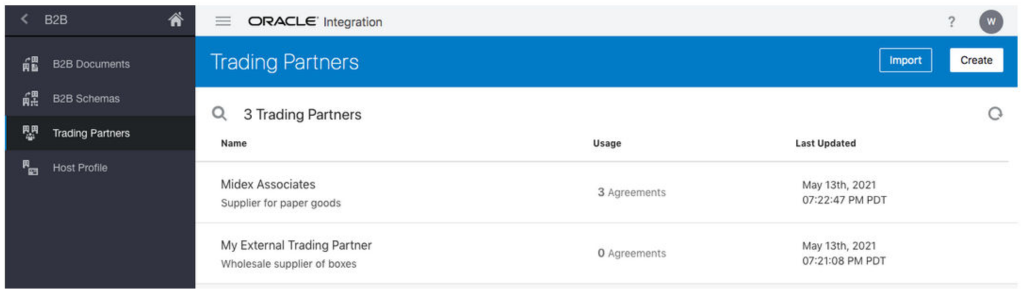
Trading Partners

A trading partner is the external business entity with which your company interacts to send or receive business documents, such as orders and invoices, in electronic form.

You select **B2B** > **Trading Partners** from the left navigation menu to access pages to register all your external trading partners and enter information on their behalf. The trading partners cannot access these pages in your Oracle Integration instance. Therefore, as the B2B system administrator, you gather information offline from your external trading partners and enter the information in these pages.

Similar to B2B for Oracle Integration, your external trading partner has a B2B application of their own. Your trading partners collect information offline about your company and enter it into their B2B application, which you cannot directly access.

Once setup is completed and tested on both ends, the two parties (your company and the external trading partner) are ready to send and receive documents.

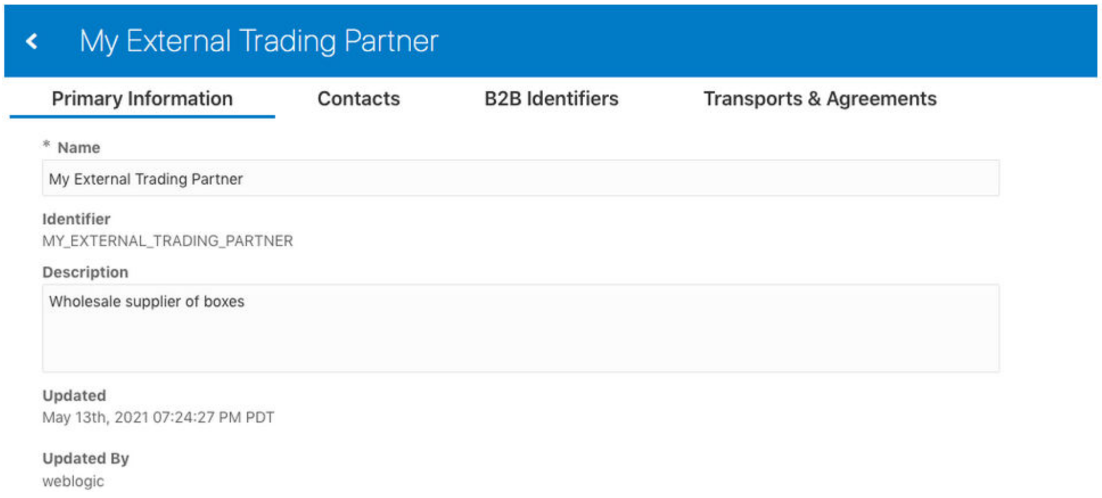


3 Trading Partners

Name	Usage	Last Updated
Midex Associates Supplier for paper goods	3 Agreements	May 13th, 2021 07:22:47 PM PDT
My External Trading Partner Wholesale supplier of boxes	0 Agreements	May 13th, 2021 07:21:08 PM PDT

You create and register an external trading partner by entering information in the following tabs, each of which are explained in subsequent sections.

- **Primary Information**
- **Contacts**
- **B2B Identifiers**
- **Transports & Agreements**



My External Trading Partner

Primary Information | Contacts | B2B Identifiers | Transports & Agreements

* Name
My External Trading Partner

Identifier
MY_EXTERNAL_TRADING_PARTNER

Description
Wholesale supplier of boxes

Updated
May 13th, 2021 07:24:27 PM PDT

Updated By
weblogic

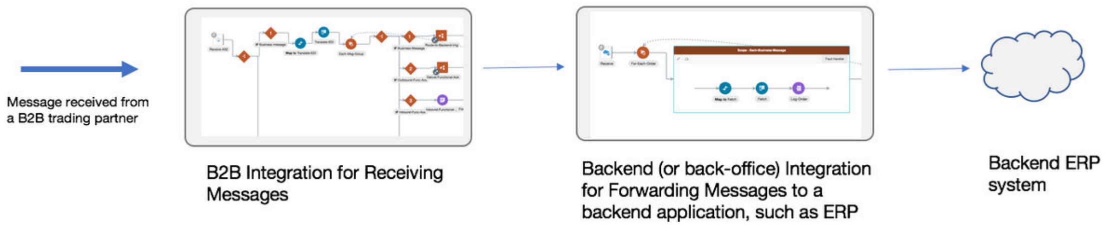
See Create Trading Partners.

Integrations Used for B2B Message Processing

The actual processing of B2B messages occurs in integrations. B2B for Oracle Integration uses a two-integration design pattern for better modularity.

Inbound Message

A B2B message received from an external trading partner is called an inbound message. The following diagram shows how an inbound message is processed through two integrations.


Message received from a B2B trading partner
B2B Integration for Receiving Messages
Backend (or back-office) Integration for Forwarding Messages to a backend application, such as ERP
Backend ERP system

In the two-integration pattern, the B2B integration for receiving messages performs the following steps:

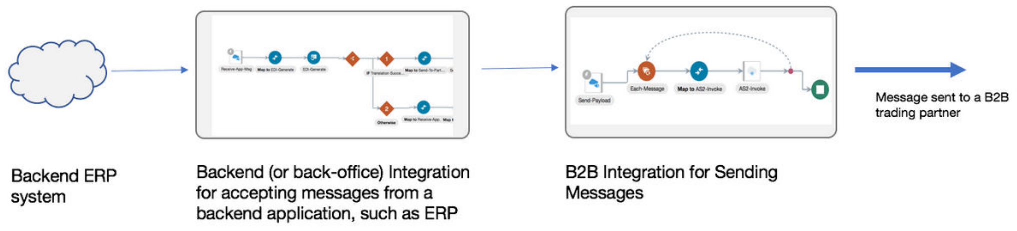
1. Receives the message from the trading partner and performs various validity checks, for example:
   - Is the message received from a known (that is, registered) trading partner? Verify this based on the authentication credentials, SSL certificates, HTTP headers, and so on.
   - Is the message signed or encrypted? If so, verify the signature and decrypt the message. This step is called unpackaging, which is similar to removing an object from its packaging.
2. Sends a transport level acknowledgment back to the trading partner, if asked by the trading partner.
3. Detects the type of payload. If it is a payload that requires translation (for example, an EDI message), parse and translate the message.

4. Sends a translator level acknowledgment, if configured.
5. The backend integration converts the message into a format that a backend application, such as ERP, can directly consume (for example, XML, JSON, CSV, and so on) and forwards the message to a backend system for further processing.

Outbound Message

A B2B message sent to an external trading partner is called as an outbound message.

The following diagram shows how an outbound message is processed through two integrations.


Backend ERP system
Backend (or back-office) Integration for accepting messages from a backend application, such as ERP
B2B Integration for Sending Messages
Message sent to a B2B trading partner

In the two-integration pattern, the backend integration performs these steps:

1. Receives an event from a backend application such as ERP for a business document that must be sent to an external trading partner.
2. Translates the message to an industry-standard B2B format (for example, Electronic Data Interchange (EDI) format).

The B2B integration for sending messages performs these steps:

1. Adds headers, encrypts, signs, and compresses the payload, as per your required configuration. This step is called packaging, which is similar to wrapping an object into an envelope and making it ready for delivery.
2. Transmits the message to the external trading partner's endpoint.
3. If the trading partner responds with a transport level acknowledgment, it updates the status of the transmitted message accordingly.

The following table summarizes the two-integration pattern for B2B message processing:

Concept	B2B Integrations for Receiving and Sending Messages	Backend Integration
Purpose	Handles the low-level technical interaction with an external trading partner, including B2B layer acknowledgments. This is provided by Oracle.	Handles the complete interaction with your backend application, such as ERP, and the message transformation between B2B canonical format and the backend application format. This is developed by your company and is typically specific to your backend application. There may be recipes and/or accelerators available for specific backend applications to get you started. See Get Started with Integration Accelerators and Recipes in *Getting Started with Oracle Integration Generation 2*.
Creation	These integrations are automatically created for each trading partner from a template. Details about how to automatically create integrations are provided. See Create B2B Integrations for Receiving and Sending.	You create this integration manually and use any available application adapters to interface with your backend application. Details about how to build this integration are provided. See Create Backend Integrations.
What does it do	• Handles packaging/ unpackaging of a message and performs receipt or transmission from or to a trading partner using an agreed-upon protocol. • Performs handshakes with the trading partner's system, generating or consuming acknowledgments as needed by a typical B2B gateway. For an inbound message, the B2B integration for receiving messages also translates the message (for example, from EDI to XML). If the inbound message contains batched transactions, those are split individually before handing them to the backend integration. However, for an outbound message, the translation is not performed in the B2B integration for sending messages. (It is performed in the backend integration.)	Handles interfacing with a backend application, such as ERP, both for the inbound and outbound sides. For an inbound message, the backend integration gets a B2B message that is already translated to a canonical format. It must convert it further and forward it to your backend application. For an outbound message, the backend integration is the entry point where an event notification from a backend application triggers this integration to perform a translation to a B2B format (for example, XML to EDI) and call the B2B integration for sending messages.

Concept	B2B Integrations for Receiving and Sending Messages	Backend Integration
How many integrations are needed	A pair of integrations are required per B2B trading partner and transport. For example, if you have 10 B2B trading partners, each with one transport, then 20 integrations are needed (one each for receiving and sending messages). These integrations are automatically created for you. See Trading Partner Mode Concepts.	A pair of integrations are required per business document type. For example, if you have three types of business documents (purchase orders, invoices, and shipment notifications), you need six integrations (three on the inbound side and three on the outbound side), regardless of the number of trading partners. Typically, you share these integrations across many trading partners as long as they use the same document format. You may actually need fewer than six integrations if some types of business documents are only used in one direction. For example, if purchase orders are only received, but never sent, you need only one backend integration for inbound purchase orders and none for outbound purchase orders. You may also need fewer integrations if you design the integrations to handle multiple document types. This is because in the simplest case, the assumption is that one backend integration handles only one type of business document. You need to create these integrations manually. See Create Backend Integrations.

Message Routing Between Integrations

The B2B integrations and backend integrations are not hard-wired to each other. They are instead dynamically wired to follow routing rules that you add declaratively. This section describes how the routing rules work using the concept of inbound and outbound agreements.

Inbound

Assume one of your trading partners is sending multiple types of documents: purchase orders and invoices. For this trading partner, one B2B integration receives messages, and it receives both types of documents. You define routing rules by adding two inbound agreements to this trading partner. When you add an inbound agreement, you must select one B2B document and one backend integration.

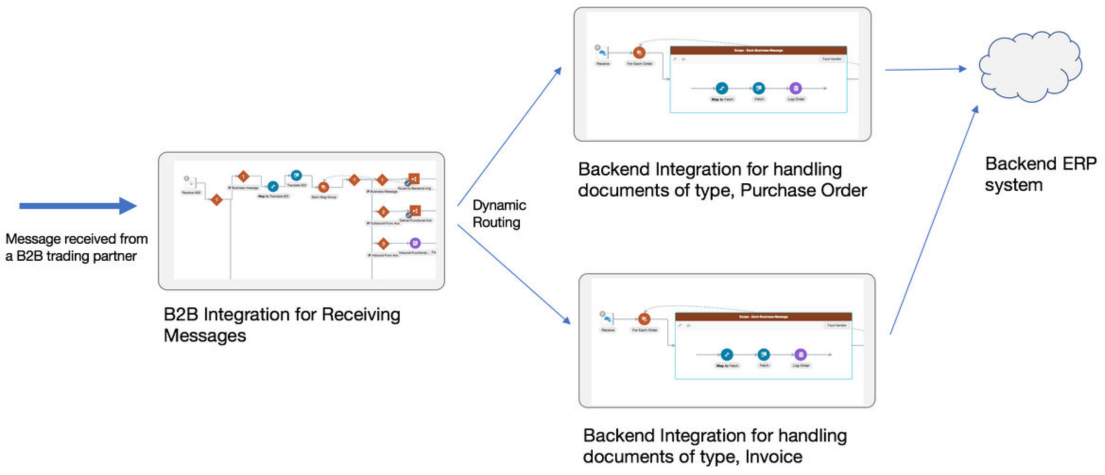
You add the following two inbound agreements:

- Inbound agreement 1 for selecting a purchase order type document and a backend integration called X.
- Inbound agreement 2 for selecting an invoice type document and a backend integration called Y.

These two agreements express the following intentions to the system:

- When a purchase order type document is received from this trading partner, it's routed to backend integration X.
- When an invoice type document is received from this trading partner, it's routed to backend integration Y.

While inbound agreements do more than routing, the routing is a primary concept. The B2B integration for receiving messages honors the routing rules at runtime and dynamically invokes backend integration X or Y, depending on the identified document type, as shown below.

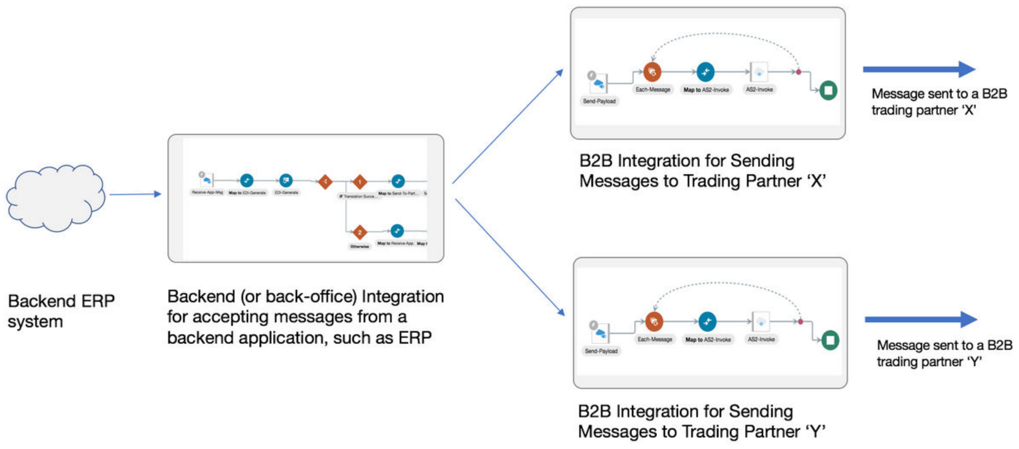

Message received from a B2B trading partner
B2B Integration for Receiving Messages
Dynamic Routing
Backend Integration for handling documents of type, Purchase Order
Backend Integration for handling documents of type, Invoice
Backend ERP system

> **Note:**
>
> You can create a single, generic backend integration that handles multiple types of documents, if you think there is enough commonality and it is easier for maintenance. In this case, select the same backend integration in the two inbound agreements.

Outbound

Assume you have two trading partners and want to route a purchase order document to one of those trading partners. In this case, the backend integration must make a routing choice to select the correct B2B integration for sending messages between two integrations.

In the outbound case, inside your backend integration, you must specify the trading partner to which you want to deliver the message. This is specified by providing a trading partner ID or an application partner ID. See Design an Outbound Backend Integration.


Backend ERP system
Backend (or back-office) Integration for accepting messages from a backend application, such as ERP
B2B Integration for Sending Messages to Trading Partner 'X'
Message sent to a B2B trading partner 'X'
B2B Integration for Sending Messages to Trading Partner 'Y'
Message sent to a B2B trading partner 'Y'

You define routing rules by adding two outbound agreements, one per trading partner. When you add an outbound agreement, you must select one B2B document and one transport. See Transports for Communication. Be aware for now that a transport maps to B2B integrations for receiving and sending messages for a given trading partner.

You add the following two outbound agreements:

- Outbound agreement 1 for selecting a purchase order type document and transport X for trading partner X.
- Outbound agreement 2 for selecting a purchase order type document and transport Y for trading partner Y.

These two agreements express the intention to the system that when a purchase order type document is sent to trading partner X, route it to the B2B integration for sending messages X. When the document is sent to trading partner Y, route it to the B2B integration for sending messages Y.

Outbound agreements do more than routing, but this section only focuses on the routing aspect.

Further concepts about agreements are provided. See Agreements.

Details are provided for creating agreements. See Create Agreements.

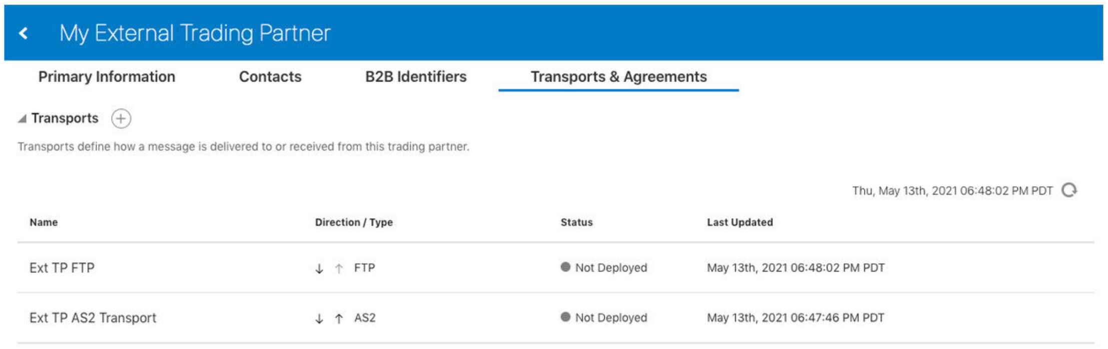
> **Note:**
>
> In the outbound case also, you can create a common backend integration that handles multiple types of documents, if that is your design preference.

Transports for Communication

Transports are configuration objects that represent a concrete communication channel to a trading partner using a specific protocol such as AS2 or FTP. You add one or more transports to a trading partner to send or receive business documents from the partner.

AS2 and FTP (which includes SFTP) are the currently supported protocols for B2B trading partner mode. If you want to use another protocol adapter in B2B for Oracle Integration, you can do so only using the standalone mode.

You define a transport for a B2B trading partner from the **B2B** > **Trading Partners** > **Transports & Agreements** tab. The following example shows a trading partner with one AS2 transport and one FTP transport.

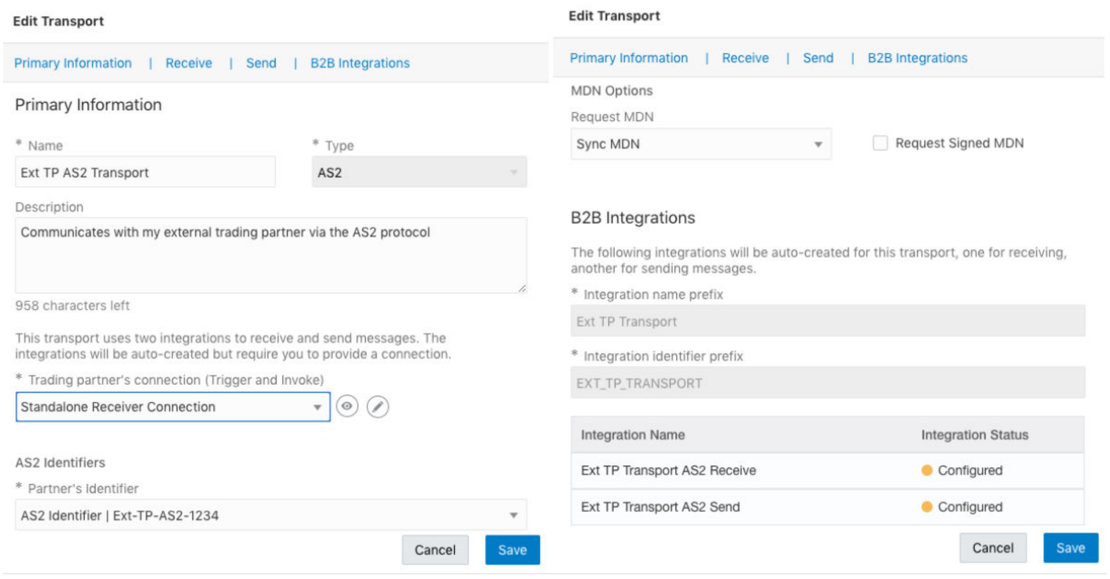

My External Trading Partner
Primary Information
Contacts
B2B Identifiers
Transports & Agreements
Transports
Transports define how a message is delivered to or received from this trading partner.
Thu, May 13th, 2021 06:48:02 PM PDT

Name	Direction / Type	Status	Last Updated
Ext TP FTP	↓ ↑ FTP	● Not Deployed	May 13th, 2021 06:48:02 PM PDT
Ext TP AS2 Transport	↓ ↑ AS2	● Not Deployed	May 13th, 2021 06:47:46 PM PDT

A transport requires the following entities to work:

- A connection that you create and configure to point to the external trading partner's endpoint. As part of this configuration, you provide the host name, port, URL, username/ password credentials, and other details that your trading partner provides to you.
- Additional configurations (for example, a directory path for FTP or an encryption algorithm for AS2).
- A pair of integrations, one for receiving messages and another for sending messages. These integrations are automatically created for you. See Integrations Used For B2B Processing.

You enter the following configuration details when you define a transport (for this example, AS2).


Edit Transport
Primary Information | Receive | Send | B2B Integrations
Primary Information
Cancel
Save
Edit Transport
Primary Information | Receive | Send | B2B Integrations
Cancel
Save

* Name: Ext TP AS2 Transport
* Type: AS2
Description: Communicates with my external trading partner via the AS2 protocol
958 characters left
This transport uses two integrations to receive and send messages. The integrations will be auto-created but require you to provide a connection.
* Trading partner's connection (Trigger and Invoke): Standalone Receiver Connection
AS2 Identifiers
* Partner's Identifier: AS2 Identifier | Ext-TP-AS2-1234

MDN Options
Request MDN: Sync MDN
Request Signed MDN: ☐

B2B Integrations
The following integrations will be auto-created for this transport, one for receiving, another for sending messages.
* Integration name prefix: Ext TP Transport
* Integration identifier prefix: EXT_TP_TRANSPORT

Integration Name	Integration Status
Ext TP Transport AS2 Receive	● Configured
Ext TP Transport AS2 Send	● Configured

The transport lifecycle consists of the following actions:

- Create a transport: Adds a definition for the transport. The pair of B2B integrations are automatically created and permanently linked to this transport.
- Deploy a transport: Makes the transport visible for runtime processing and also activates the B2B integrations. This transport can now receive and send messages.

- Redeploy a transport: Applies configuration changes to the runtime on-the-fly without disrupting message processing (with some restrictions).
- Undeploy a transport: Hides the transport from runtime processing and also deactivates the B2B integrations. This transport can no longer receive and send messages.
- Delete a transport: Removes the definition and also deletes the B2B integrations.

See Define Transports.

AS2

Applicability Statement 2 (AS2) is an HTTP-based protocol designed for B2B that adds a comprehensive set of data security features around data confidentiality, data integrity/authenticity, and nonrepudiation.

The AS2 specification is covered by RFC 4130. B2B for Oracle Integration supports AS2 versions 1.0 and 1.1.

An AS2 transport offers configuration options specific to AS2 that work in conjunction with the AS2 connection and the certificate management user interface.

For example, if you want to sign and encrypt the outbound messages:

1. Use the Oracle Integration certificate management capabilities by uploading your certificate under **Settings** > **Certificates**.
2. Enter the signing and encryption certificate alias in the AS2 Adapter connection selected in the AS2 transport.
3. Select an encryption and signing algorithm in the AS2 transport configuration.

The simplest AS2 communication uses no encryption, signing, and compression. If you are learning about AS2, you can start simple and add the security layers later.

There is the concept of an electronic read receipt in AS2, officially known as Message Disposition Notification (MDN). It is a transport level acknowledgment used as a confirmation that the other party has received your message intact. B2B for Oracle Integration generates and consumes MDN messages (when enabled) and correlates them to the original transmissions. The Track B2B Messages page, described later, enables you to view the AS2 messages and MDN acknowledgments.

More details about the AS2 transport are provided. See Define Transports.

FTP

File Transfer Protocol (FTP) and Secure File Transfer Protocol (SFTP) are commonly used for B2B communications. An FTP transport also works in conjunction with an FTP Adapter connection.

In an FTP Adapter connection, you specify the hostname, port, credentials, and other security configurations. In the FTP transport, you enter the input and output directories, file name pattern, and other details.

One important aspect of an FTP transport is that the receiving side polls the input directory on a time-based schedule. You can set up the schedule by selecting the **Receive Schedule** option of the actions menu for an FTP transport.

More details about the FTP transport are provided. See Define Transports.

Agreements

You define one or more agreements for a B2B trading partner with an intent to send or receive only certain types of business documents to or from that trading partner. An agreement literally means that your company and the external trading partner have formally agreed upon the terms for the exchange of specific business documents.

An agreement has the following purposes:

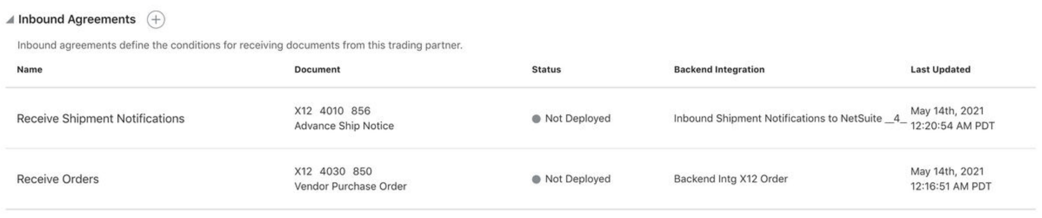
- Armed with the knowledge of which documents to expect from a given trading partner, B2B only accepts one of the agreed-upon documents. Any unexpected document type is rejected both while receiving from or sending to that trading partner.
- Defines the data format for the documents exchanged. For example, for an EDI document, syntax validations and parsing is done based on the B2B document selected in an agreement. Both parties, your company and the external trading partner, must decide in advance on the data format to use for interoperability to work. One of the parties typically shares an implementation guide containing the data format definition and the other party complies with it and creates an equivalent data format definition on their side.
- Defines behavior for message processing. For example, syntax validations on the data format can be turned on or off in an agreement, among other settings.
- Defines rules for the routing of documents. For example, when receiving documents, an inbound agreement defines the backend integration to which to route a document, based on its type. While sending documents, an outbound agreement defines which B2B sending integration to hand over a specific document for delivery to a trading partner.

The agreement lifecycle consists of the following actions:

- Create an agreement: Adds the definition in the design-time only (but unless deployed, the new agreement is not enforced at runtime).
- Deploy and redeploy an agreement: Makes the agreement visible for runtime processing and is immediately enforced.
- Undeploy an agreement: Hides the agreement from runtime processing, making it no longer effective, starting immediately.
- Delete an agreement: Removes it from design time.

The following examples show inbound and outbound agreements defined for a trading partner:

- Inbound agreement:

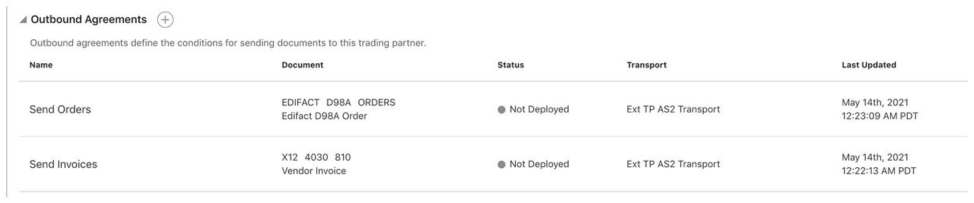


Inbound Agreements

Inbound agreements define the conditions for receiving documents from this trading partner.

Name	Document	Status	Backend Integration	Last Updated
Receive Shipment Notifications	X12 4010 856 Advance Ship Notice	● Not Deployed	Inbound Shipment Notifications to NetSuite __4_	May 14th, 2021 12:20:54 AM PDT
Receive Orders	X12 4030 850 Vendor Purchase Order	● Not Deployed	Backend Intg X12 Order	May 14th, 2021 12:16:51 AM PDT

- Outbound agreement:



Outbound Agreements

Outbound agreements define the conditions for sending documents to this trading partner.

Name	Document	Status	Transport	Last Updated
Send Orders	EDIFACT D98A ORDERS Edifact D98A Order	● Not Deployed	Ext TP AS2 Transport	May 14th, 2021 12:23:09 AM PDT
Send Invoices	X12 4030 810 Vendor Invoice	● Not Deployed	Ext TP AS2 Transport	May 14th, 2021 12:22:13 AM PDT

See Create Agreements.

B2B Documents and Schemas

A B2B document is a mandatory object required by an agreement that specifies a data format and some additional configuration pertaining to the data format. A data format is specified using the following properties.

- Document standard: The X12 and EDIFACT EDI standards are currently supported.
- Document version: A version as defined by the standards body.
- Document type: A type as defined by the standards body.
- Document schema: A standard or customized variant defined in a B2B schema object. Standard means the pristine schema defined in the data dictionary published by the standards body.

Additional configuration enables one or more business identifiers for the B2B runtime to be extracted and displayed in the Track B2B Messages page. This page is accessible in the left navigation pane from **Monitoring** > **B2B Tracking** > **Business Messages**.

A B2B schema is an optional object that represents a customized variant of one of the standard schemas.

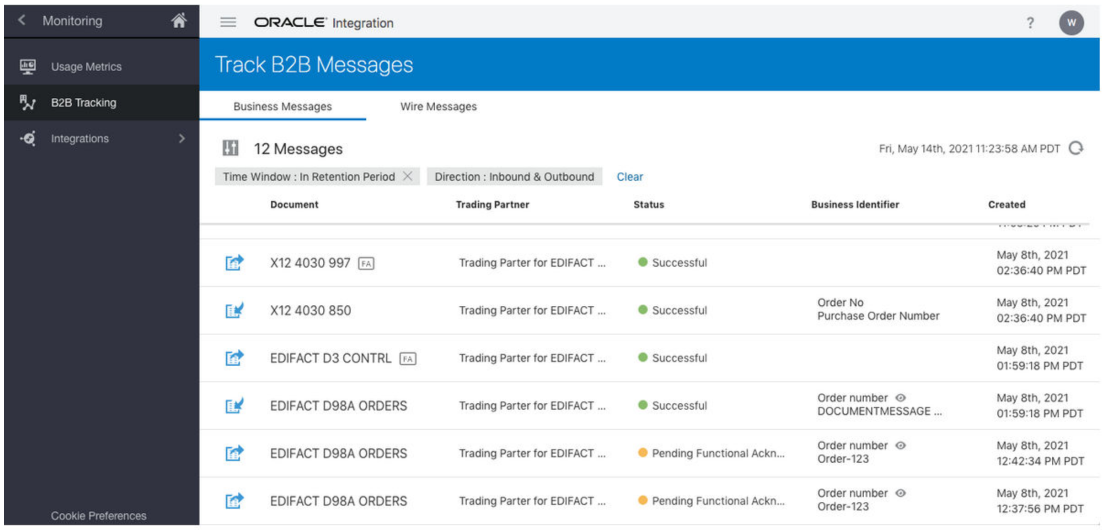
Additional details about B2B documents and B2B schemas are provided. See B2B Documents and schemas.

Additional details about agreements are provided. See Create Agreements.

Message Persistence and Tracking

As messages flow through the B2B integrations for receiving and sending messages, each inbound and outbound message is persisted separately, in addition to the usual integration instance tracking.

The persisted B2B messages can be viewed from a specialized Track B2B Messages page accessible from the left navigation menu under **Monitoring** > **B2B Tracking**.



Document	Trading Partner	Status	Business Identifier	Created
X12 4030 997 FA	Trading Parter for EDIFACT ...	● Successful		May 8th, 2021 02:36:40 PM PDT
X12 4030 850	Trading Parter for EDIFACT ...	● Successful	Order No Purchase Order Number	May 8th, 2021 02:36:40 PM PDT
EDIFACT D3 CONTRL FA	Trading Parter for EDIFACT ...	● Successful		May 8th, 2021 01:59:18 PM PDT
EDIFACT D98A ORDERS	Trading Parter for EDIFACT ...	● Successful	Order number DOCUMENTMESSAGE ...	May 8th, 2021 01:59:18 PM PDT
EDIFACT D98A ORDERS	Trading Parter for EDIFACT ...	● Pending Functional Ackn...	Order number Order-123	May 8th, 2021 12:42:34 PM PDT
EDIFACT D98A ORDERS	Trading Parter for EDIFACT ...	● Pending Functional Ackn...	Order number Order-123	May 8th, 2021 12:37:56 PM PDT

Business Messages

In this view, you see B2B messages as individual business transactions. Rows marked

FA

denote functional acknowledgments. Assume you receive an inbound message containing batched transactions. This view shows individual rows for each individual transaction within that batch.

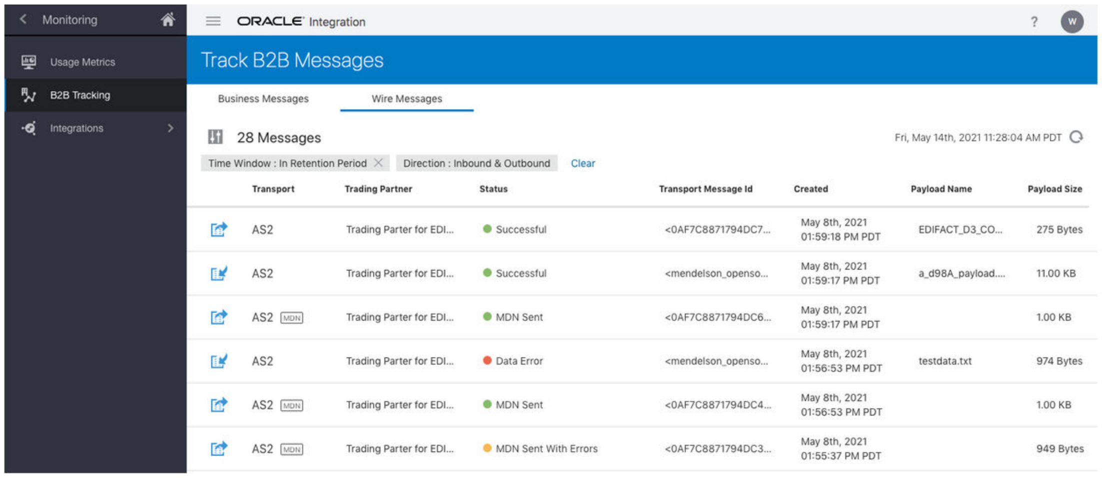
Wire Messages

An alternate, low-level technical view of B2B messages is provided with the wire messages. In this view, you see messages in the form in which they are transmitted to or received from a trading partner. This view is useful for troubleshooting, in case the message failed to be delivered to a trading partner or a received message failed signature verification. Rows marked with

MDN

denote AS2 MDNs (electronic return receipts).

In the case where you receive an inbound message containing a batched transaction, you only see one row corresponding to the actual batch message that was received.



Transport	Trading Partner	Status	Transport Message Id	Created	Payload Name	Payload Size
AS2	Trading Parter for EDI...	● Successful	<0AF7C8871794DC7...	May 8th, 2021 01:59:18 PM PDT	EDIFACT_D3_CO...	275 Bytes
AS2	Trading Parter for EDI...	● Successful	<mendelson_openso...	May 8th, 2021 01:59:17 PM PDT	a_d98A_payload....	11.00 KB
AS2 MDN	Trading Parter for EDI...	● MDN Sent	<0AF7C8871794DC6...	May 8th, 2021 01:59:17 PM PDT		1.00 KB
AS2	Trading Parter for EDI...	● Data Error	<mendelson_openso...	May 8th, 2021 01:56:53 PM PDT	testdata.txt	974 Bytes
AS2 MDN	Trading Parter for EDI...	● MDN Sent	<0AF7C8871794DC4...	May 8th, 2021 01:56:53 PM PDT		1.00 KB
AS2 MDN	Trading Parter for EDI...	● MDN Sent With Errors	<0AF7C8871794DC3...	May 8th, 2021 01:55:37 PM PDT		949 Bytes

More B2B tracking details are provided. See B2B Tracking.

Manage B2B Trading Partners

This section describes how to configure trading partners in B2B for Oracle Integration.

Note:

This feature is only available on Oracle Integration Generation 2.

- Define the Host Profile
- Create Trading Partners
- Define Transports
- Create Agreements
- Export and Import a Trading Partner

Trading partner concepts are provided. See Trading Partners.

Define the Host Profile

This section describes how to define the host profile (host company) in B2B for Oracle Integration.

Host trading partner concepts are provided. See Host Company.

- Define Identifiers in the Host Profile
- Create the Host Profile

Define Identifiers in the Host Profile

You specify the host identifier and value when configuring the host profile. Understand the following details about how the identifier is used.

- Where the identifiers are used:
  The following table lists where each type of identifier is used and its purpose.

Identifier Type	Where Used	Purpose	Value Restrictions
AS2 Identifier	**Trading Partners** > **Transports & Agreements** > **AS2** > **AS2 Identifiers** > **Host Identifier**	Mandatory only when the AS2 transport protocol is used. This identifier type is not used for the FTP transport protocol. For an outbound message, this value is inserted as the `AS2-From` HTTP header of the AS2 message. For an inbound message, this value is used for validation against the `AS2-To` header of the incoming message.	Up to 128 printable ASCII characters except double quotes or back slashes. This value can be any value agreed upon with the trading partner as a value with which they can identify your company. The value is case sensitive.
EDI Interchange ID	**Trading Partner** > **Transports & Agreements** > **Outbound Agreement** > **Select Host Identifiers**	Mandatory for all EDI data formats. This identifier is used as the Interchange Sender ID field of the interchange envelope (ISA segment for X12 and UNB segment for EDIFACT). For an outbound message, this value is inserted as the Interchange Sender ID. For an inbound message, this identifier (from the host profile) is not used.	Up to 15 characters for X12. Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any delimiter characters.

Identifier Type	Where Used	Purpose	Value Restrictions
EDI Interchange ID Qualifier	Same as above.	Mandatory for X12 and optional for EDIFACT. This identifier is used as the Interchange Sender ID Qualifier field of the interchange envelope of the EDI payload. It's a code to indicate the category of the value specified in the EDI Interchange ID (for example, DUNS number, IATA number, and so on). For an outbound message, this value is inserted as the Interchange Sender ID Qualifier. For an inbound message, the value (from the host profile) is not used.	Must be exactly two characters for X12. The value must be from an X12 code list. See EDI Standards Reference. A generic sample value for X12 is `ZZ`. Up to four characters for EDIFACT. The value must be from an EDIFACT code list (https://www.gefeg.com/jswg/cl/v3/11b/cl3.htm). A generic sample value for EDIFACT is `ZZZ`. The value is case sensitive and must not contain any delimiter characters.
EDI Interchange Internal ID	Same as above.	Only used for EDIFACT. In that, it is also optional. EDIFACT defines this field as an identification (for example, a division, branch, or computer system/process) specified by the sender of the interchange to be included if agreed to by the recipient in response interchanges to facilitate internal routing.	Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any delimiter characters.
EDI Interchange Internal Sub ID	Same as above.	Only used for EDIFACT syntax version 4. In that, it is also optional. EDIFACT defines this field as the sublevel of sender internal identification when further sublevel identification is required.	Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any of the delimiter characters.

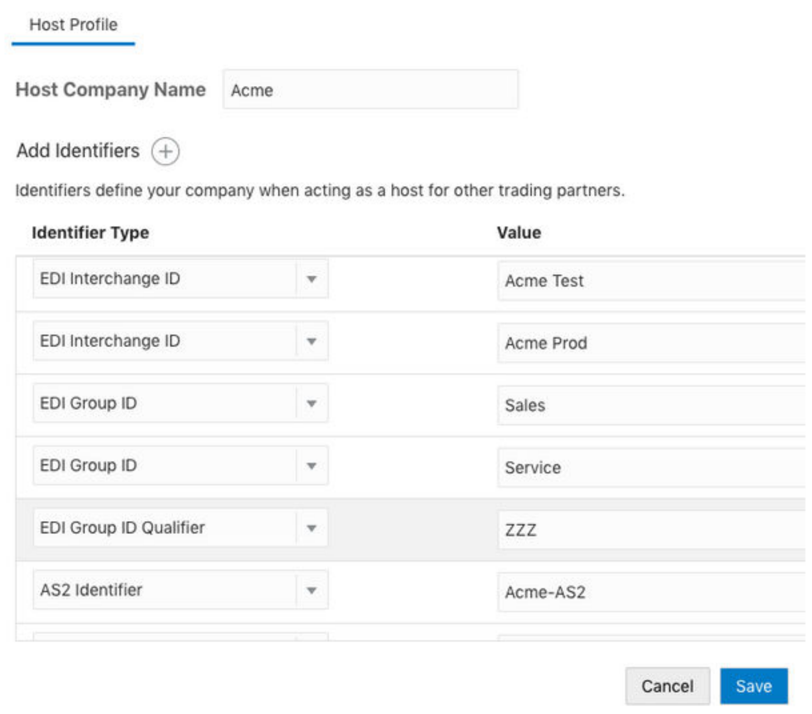
Identifier Type	Where Used	Purpose	Value Restrictions
EDI Group ID	Same as above.	Mandatory for X12 and optional for EDIFACT. EDIFACT defines this field as an identification of the sender's division, department, and so on from which a group of messages is sent. X12 has a similar definition. For an outbound X12 message, this value is inserted in the GS segment as the Application Sender's Code. For an outbound EDIFACT message, this value is inserted in the UNG segment as the sender identification (a UNG is only generated when this identifier is selected in an outbound agreement; otherwise not). For an inbound message, the value from the host profile is not used.	Minimum of two characters and up to 15 characters for X12. Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any delimiter characters.
EDI Group ID Qualifier	Same as above.	Only used for EDIFACT. In that, it is also optional. It's a code to indicate the category of the value specified in the EDI Group ID (for example, DUNS number, IATA number, and so on).	Up to four characters for EDIFACT. The value must be from an EDIFACT code list (https://www.gefeg.com/jswg/cl/v3/11b/cl3.htm). A generic sample value is `ZZZ`. The value is case sensitive and must not contain any delimiter characters.
Application Partner ID	Not used.	This identifier from the host profile is not used.	Not used.

- Host identifiers in use are protected:
  You cannot delete host identifiers referenced in transports or agreements. The associations must be removed from the transports or agreements *before* you can delete the host identifier from the host profile page.
- Multiple identifiers of the same type:
  You may add multiple identifiers of the same identifier type. For example, you may want to add identifiers based on X12 or EDIFACT usage, based on business units within your company, or based on an environment such as development, test, production, and so on.

The combination of identifier type and value must be unique, and this validation is enforced in the user interface.

Even though you may define multiple identifiers of the same type, in outbound agreements, you must choose specific unique types so that B2B for Oracle Integration knows exactly which identifiers to insert in an outbound message without ambiguity.

In the following example, two different rows, both with EDI Interchange ID and EDI Group ID, are defined.



Host Company Name: Acme

Identifiers define your company when acting as a host for other trading partners.

Identifier Type	Value
EDI Interchange ID	Acme Test
EDI Interchange ID	Acme Prod
EDI Group ID	Sales
EDI Group ID	Service
EDI Group ID Qualifier	ZZZ
AS2 Identifier	Acme-AS2

- Updating values and applying changes to runtime:
  You can update existing identifier values any time. However, they are not used in runtime B2B processing until:
  - Each transport where an identifier value was referenced is redeployed.
  - Each outbound agreement where an identifier value was referenced is redeployed.

  Redeployment is a lifecycle action available for transports and agreements. It applies changes to the runtime on-the-fly, without disruption of message processing.

Create the Host Profile

1. In the left navigation pane, click **B2B** > **Host Profile**.
2. In the **Host Company Name** field, enter your company name. The name is currently only for reference and not used elsewhere.
3. Select a host identifier or, if none are defined, click ⊕

. You add identifiers to the host profile on behalf of your company. This is typically a one-time activity that you perform before adding your first trading partner.
Host identifiers define your company when acting as the host interacting with other trading partners. They identify and validate the source of the document when sent by the host. Identifiers defined here are used in two places:

- Transports
- Outbound agreements

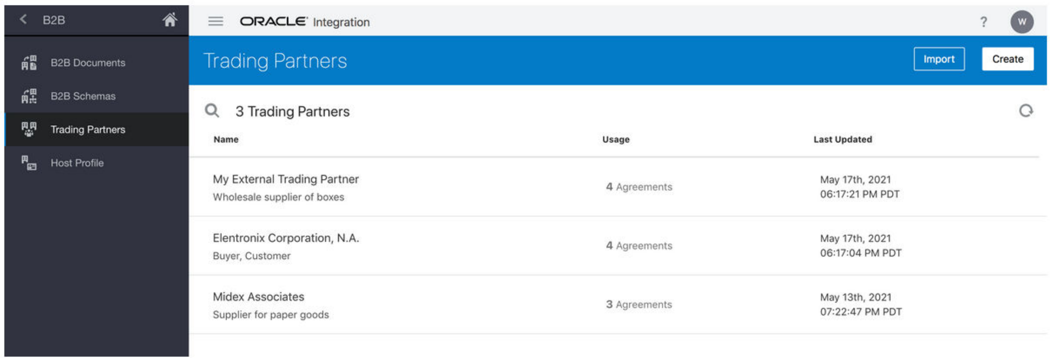
4. Select a host identifier name and specify a value. You can specify multiple name and value pairs for the host.
5. Click **Save**.
   If you change the host identifier value used in a deployed agreement, the changes only take effect after you explicitly redeploy the agreement between the host and the trading partner.
6. If you want to delete any unneeded host identifiers, click
   . You cannot delete host identifiers referenced by active agreements. The associations must be removed from the agreements before you can delete the host identifier from the host profile page.

Create Trading Partners

You can create and manage trading partners. A trading partner is the external business entity with which your company interacts to send or receive business documents, such as orders and invoices, in electronic form.

Trading partner concepts are provided. See Trading Partners.

1. In the left navigation pane, click **B2B** > **Trading Partners**.
   The Trading Partners page is displayed. The **Usage** column shows the number of agreements associated with the trading partner. Click the entry to show the number of inbound and outbound agreements for that trading partner.



3 Trading Partners

Name	Usage	Last Updated
My External Trading Partner Wholesale supplier of boxes	4 Agreements	May 17th, 2021 06:17:21 PM PDT
Elentronix Corporation, N.A. Buyer, Customer	4 Agreements	May 17th, 2021 06:17:04 PM PDT
Midex Associates Supplier for paper goods	3 Agreements	May 13th, 2021 07:22:47 PM PDT

2. Click **Create**.
3. Enter the trading partner name and an optional description. The **Identifier** field is automatically populated with the name you enter. The values for both must be unique.

Note:

The identifier cannot be changed after creation.

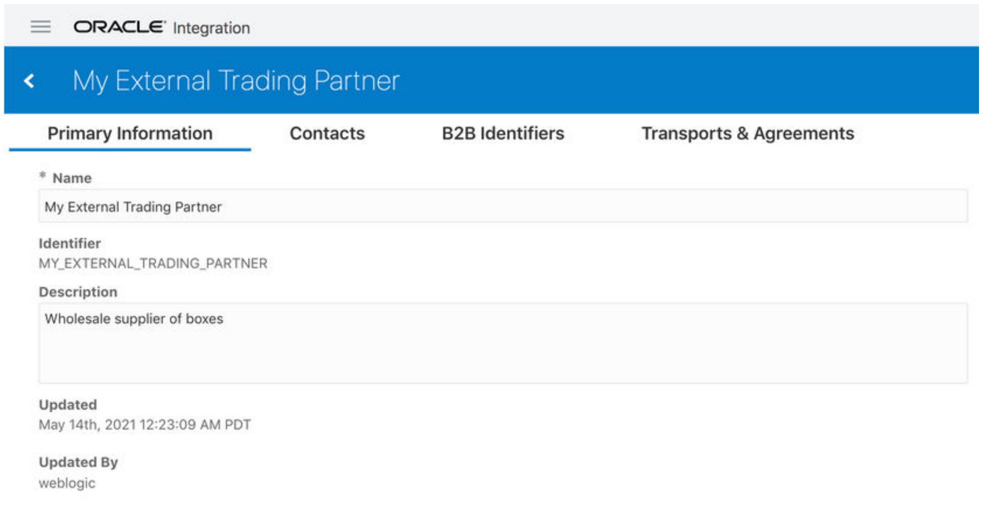
4. **Create**.
   A details panel is displayed that enables you to configure additional information about the newly created trading partner.

See the following sections:

- View Primary Information
- Select Contacts
- Define B2B Identifiers

View Primary Information

1. Click **Primary Information** to view the same information you entered when you created the new trading partner. Both the name and identifier of a trading partner must be unique across all trading partners.



* Name
My External Trading Partner

Identifier
MY_EXTERNAL_TRADING_PARTNER

Description
Wholesale supplier of boxes

Updated
May 14th, 2021 12:23:09 AM PDT

Updated By
weblogic

2. If needed, change the name of the trading partner.
   The trading partner name can be changed at any time. There are valid scenarios for renaming an actively-used trading partner. In the real world, companies change their names, have mergers and acquisitions, sell business units, and so on. You need the ability to reflect the new name here.

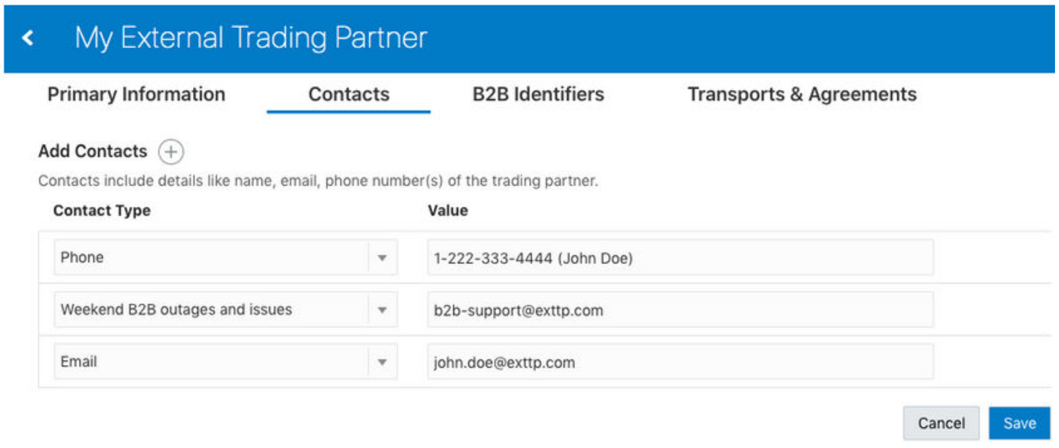
   Implications of renaming an active trading partner:

   The trading partner's name is displayed in a column of the Track B2B Messages page for messages received from or sent to this trading partner. If you change the name of an actively-used trading partner with one or more runtime messages processed, then any existing messages on the Track B2B Messages page still show the old name of the trading partner prior to the renaming. Any new runtime messages record the new name of the trading partner. While this is intentionally done for auditing purposes, when you filter messages by a trading partner, it shows all messages (both old and new) for that trading partner correctly. Old messages show the old name and new messages show the new name.

Select Contacts

You can add ways to contact the trading partner, such as their name, email, phone number, or short message service (SMS) number. The **Contact Type** and **Value** fields are both free text fields. This enables you to enter custom text. Use this information to contact individuals offline, as needed. The **Contacts** field is currently provided only for reference and is not used in B2B for Oracle Integration.

1. Click **Contacts**.
2. From the **Contact Type** list, select a method for contacting the trading partner, such as their name, email, phone number, or SMS number. Use this information to contact individuals offline as needed, and not from within B2B for Oracle Integration.
3. Add a corresponding value, then click **Save**.



Contacts include details like name, email, phone number(s) of the trading partner.

Contact Type	Value
Phone	1-222-333-4444 (John Doe)
Weekend B2B outages and issues	b2b-support@exttp.com
Email	john.doe@exttp.com

Define B2B Identifiers

You collect identifying information from your external trading partner and enter it on their behalf in the **B2B Identifiers** section. This is very similar, in concept, to the identifiers specified under **B2B** > **Host Profile** > **Identifiers**. In a message exchange, the host's identifiers and trading partner's identifiers are used in the role of a sender or receiver, depending on the message direction.

Direction	Sender	Receiver
Inbound message	Trading Partner (The trading partner's identifiers are used as **Sender ID** or **From Party**.)	Your company (that is, the host) (Host identifiers are used as **Receiver ID** or **To Party**.)
Outbound message	Your Company (that is, the host) (Host identifiers are used as **Sender ID** or **From Party**.)	Trading Partner (The trading partner's identifiers are used as **Receiver ID** or **To Party**.)

B2B Identifiers defined here are used in two places:

- Transports
- Outbound agreements

Understand the following details about how the identifier is used.

- The following table lists where each type of identifier is used and its purpose.

Identifier Type	Where Used	Purpose	Value Restrictions
AS2 Identifier	**Trading Partners** > **Transports & Agreements** > **AS2** > **AS2 Identifiers** > **Partner's Identifier**	Mandatory only when the AS2 transport protocol is used. This identifier type is not used for the FTP transport protocol. For an outbound message, this value is inserted as the `AS2-To` HTTP header of the AS2 message. For an inbound message, this value is used for validation against the `AS2-From` header of the incoming message.	Up to 128 printable ASCII characters except double quotes or backslashes. It can be any value agreed upon with the trading partner as a value with which you can identify the trading partner. The value is case sensitive.
EDI Interchange ID	**Trading Partner** > **Outbound Agreement** > **Select Trading Partner Identifiers** Also used implicitly for all inbound agreements (** see the note)	Mandatory for all EDI data formats. This identifier is used as either the Interchange Sender or Receiver ID field of the interchange envelope (ISA segment for X12, and UNB segment for EDIFACT). For an outbound message, this value is inserted as the Interchange Receiver ID. For an inbound message, this identifier is used as the Interchange Sender ID to identity a trading partner as a sender of a message. If the EDI Interchange ID on its own does not uniquely identify a trading partner, it is used in combination with the EDI Group ID to locate a unique trading partner.	Up to 15 characters for X12. Up to 35 characters for EDIFACT. The value is case-sensitive and must not contain any of the delimiter characters.

Identifier Type	Where Used	Purpose	Value Restrictions
EDI Interchange ID Qualifier	Same as above.	Mandatory for X12 and optional for EDIFACT. This identifier is used as the Interchange Sender or Receiver ID Qualifier field of the interchange envelope of the EDI payload. It is a code to indicate the category of the value specified in the EDI Interchange ID (for example, DUNS number, IATA number, and so on). For an outbound message, this value is inserted as the Interchange Receiver ID Qualifier. For an inbound message, the value is not currently used (if it were used, it would be treated as the Interchange Sender ID Qualifier).	Must be exactly two characters for X12. The value must be from an X12 code list. See EDI Standards Reference. A generic sample value for X12 is `ZZ`. Up to four characters for EDIFACT. The value must be from an EDIFACT code list (https://www.gefeg.com/jswg/cl/v3/11b/cl3.htm). A generic sample value for EDIFACT is `ZZZ`. The value is case sensitive and must not contain any delimiter characters.

Identifier Type	Where Used	Purpose	Value Restrictions
EDI Interchange Internal ID	Same as above.	Only used for EDIFACT, and in that too, it is optional. EDIFACT defines this field as an identification (for example, a division, branch or computer system/process) specified by the sender of the interchange to be included if agreed by the recipient in response interchanges to facilitate internal routing. For an outbound message, this value is inserted as the EDI Interchange Internal ID field in the EDIFACT UNB envelope. For an inbound message, this value is not used.	Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any delimiter characters.
EDI Interchange Internal Sub ID	Same as above.	Only used for EDIFACT syntax version 4. In that, it is also optional. EDIFACT defines this field as the sublevel of sender internal identification when further sublevel identification is required. For an outbound message, this value is inserted as the EDI Interchange Internal Sub ID field in the EDIFACT UNB envelope, only if the message follows the Syntax Version 4 (which is the default). For an inbound message, this value is not used.	Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any of the delimiter characters.

Identifier Type	Where Used	Purpose	Value Restrictions
EDI Group ID	Same as above.	This is mandatory for X12 and optional for EDIFACT. EDIFACT defines this field as an identification of the sender's division, department, and so on from which a group of messages is sent. X12 has a similar definition. For an outbound X12 message, this value is inserted in the GS segment as the Application Receiver's Code. For an outbound EDIFACT message, this value is inserted in the UNG segment as the Receiver's Identification (a UNG is only generated when this identifier is selected in an outbound agreement; otherwise, not). For an inbound message, this value is used only in case the EDI Interchange ID, on its own, is not enough to uniquely identify a trading partner. In that case, a combination of the EDI Interchange ID and the EDI Group ID is used to locate a unique trading partner.	Minimum of two characters and up to 15 characters for X12. Up to 35 characters for EDIFACT. The value is case sensitive and must not contain any delimiter characters.

Identifier Type	Where Used	Purpose	Value Restrictions
EDI Group ID Qualifier	Same as above.	Only used for EDIFACT. In that, it is optional. It is a code to indicate the category of the value specified in the EDI Group ID (for example, DUNS number, IATA number, and so on). Only used for an outbound message, to insert as EDI Group ID Qualifier, if specified.	Up to four characters for EDIFACT. The value must be from an EDIFACT code list (https://www.gefeg.com/jswg/cl/v3/11b/cl3.htm). A generic sample value is `ZZZ`. The value is case sensitive and must not contain any delimiter characters.
Application Partner ID	Implicitly used by all outbound agreements (** see the note)	Optionally used as an alternate way to specify which trading partner to which to route an outbound message. For an outbound message, you can specify either a Trading Partner ID or an Application Partner ID for trading partner identification. See Create Backend Integrations for more details on this usage.	N/A

Note:

** An implicit usage means you don't explicitly select the identifier in an inbound or outbound agreement. Instead, when you deploy an agreement, the identifier is automatically used in the runtime processing.

- Used B2B identifiers are protected:
  You cannot delete trading partner's B2B identifiers that are explicitly referenced in transports or agreements. The associations must be removed from the transports or agreements *before* you can delete an identifier from the trading partner's **B2B Identifiers** section.
- Multiple identifiers of the same type:
  The concept is similar to defining multiple identifiers in the host profile.

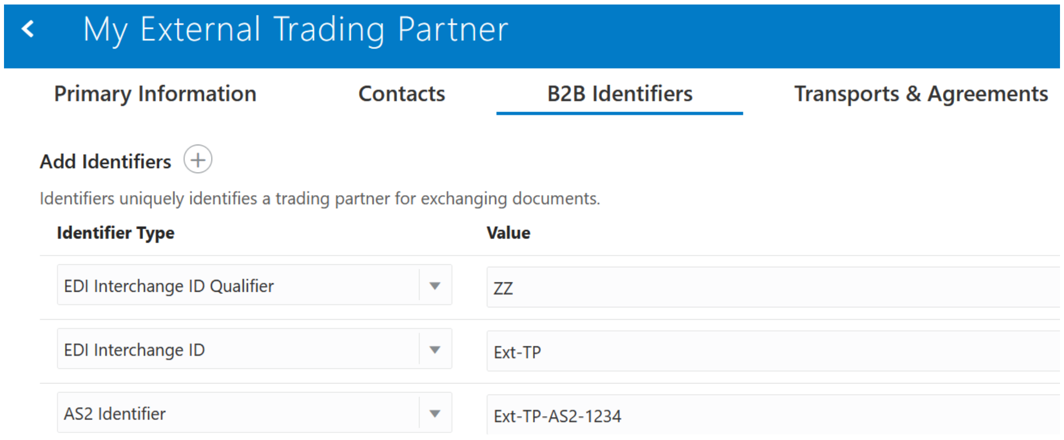
  You may add multiple B2B identifiers of the same identifier type. For example, you may want to add identifiers based on X12 or EDIFACT usage, based on business units within the trading partner, or based on an environment such as development, test, production, and so on. The combination of identifier type and value must be unique. This validation is enforced in the user interface.

  Even though you may define multiple identifiers of the same type, in outbound agreements, you must select specific unique types so that B2B knows exactly which identifiers to insert in an outbound message without ambiguity.

- Updating values and applying changes to runtime:
  You can update existing trading partner's B2B identifier values at any time. However, they are not used in runtime B2B processing, until:
  - Each transport where an identifier value was referenced is redeployed.
  - Each outbound agreement where an identifier value was referenced is redeployed.
  - For B2B identifiers that are implicitly used, any of the agreements is redeployed (it can be any one of the inbound or outbound agreements).

  Redeployment is a lifecycle action available for transports and agreements. It applies changes to the runtime, on-the-fly, without disruption of message processing.

1. Click **B2B Identifiers** to define the identifiers that uniquely identify a trading partner. This information is similar to what you defined for the host. See Define the Host Profile.
2. Select an identifier name and specify a value. You can specify multiple name and value pairs. If no identifiers are defined, click
   ⊕
   to add a new identifier, selecting an identifier type and entering a value.



Identifiers uniquely identifies a trading partner for exchanging documents.

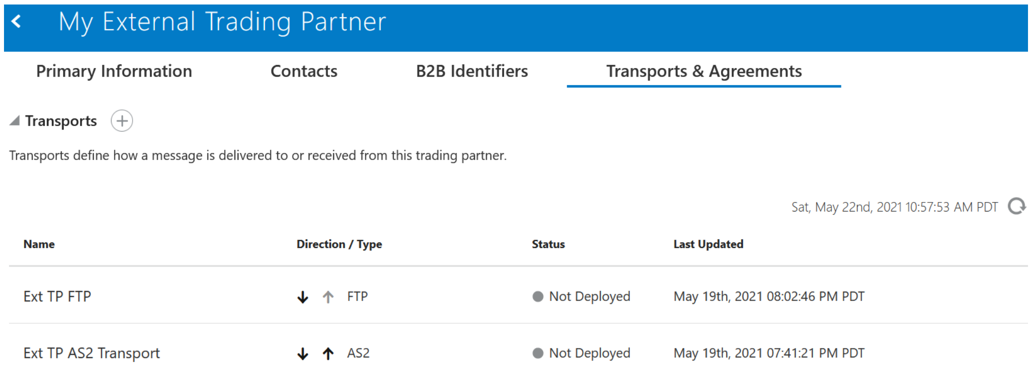
Identifier Type	Value
EDI Interchange ID Qualifier	ZZ
EDI Interchange ID	Ext-TP
AS2 Identifier	Ext-TP-AS2-1234

3. Click **Save**.
4. Click
   <
   to return to the Trading Partners page.

Define Transports

A transport maps to a technical communication protocol. In most cases, you add one transport per trading partner to receive or send messages from the partner. The AS2 and FTP transport protocols are currently supported.

The Transports & Agreements page shows a list of transports for a trading partner.



Transports define how a message is delivered to or received from this trading partner.

Sat, May 22nd, 2021 10:57:53 AM PDT

Name	Direction / Type	Status	Last Updated
Ext TP FTP	↓ ↑ FTP	● Not Deployed	May 19th, 2021 08:02:46 PM PDT
Ext TP AS2 Transport	↓ ↑ AS2	● Not Deployed	May 19th, 2021 07:41:21 PM PDT

Each transport is listed with its name, direction and type, status, and last updated time. The direction is an indicator of whether it is configured to receive (down arrow), send (up arrow), or both. Status can be:

- Not deployed
- Deploying
- Deployed
- Failed

Hover the cursor over a row to see additional icon buttons to view configuration, edit configuration, and open an action menu. When you define the transport protocol (AS2 or FTP) and transport parameters to use, you automatically create two integrations (to receive and send messages from or to the trading partner).

A transport includes the following configuration settings:

- A mandatory connection that you must separately create and select in the transport
- Protocol-specific settings (for example, AS2 settings, FTP settings)

Transport concepts are provided. See Transports for Communication.

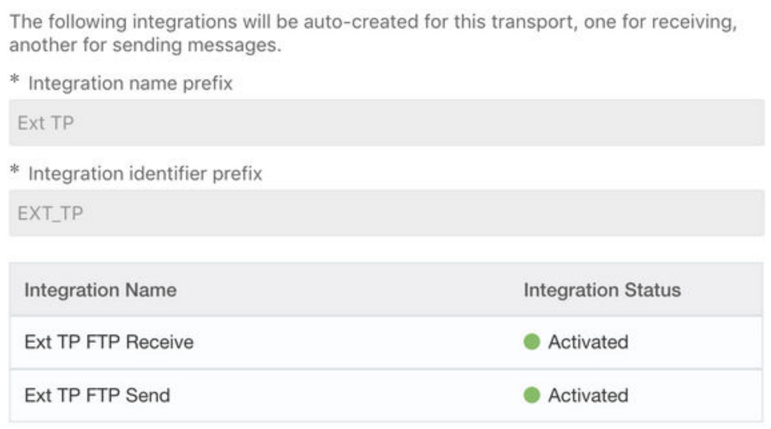
B2B Integrations

Two integrations are created automatically under-the-covers when a transport is created. These integrations are the heart of the transport for its runtime functioning. The two integrations actually process the runtime messages that pass through the transport.

- B2B integration for receiving messages
- B2B integration for sending messages

See Integrations Used for B2B Message Processing.

The **B2B Integrations** section of the Transport details panel shows the status of the two integrations.


The following integrations will be auto-created for this transport, one for receiving, another for sending messages.

* Integration name prefix

Ext TP

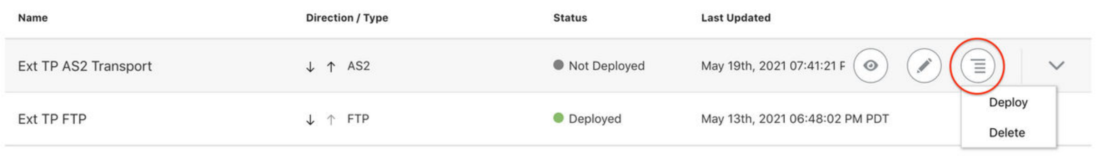
* Integration identifier prefix

EXT_TP

Integration Name	Integration Status
Ext TP FTP Receive	● Activated
Ext TP FTP Send	● Activated

Lifecycle Actions for Transports

Click the action menu on a row to view available actions.



Name	Direction / Type	Status	Last Updated
Ext TP AS2 Transport	↓ ↑ AS2	● Not Deployed	May 19th, 2021 07:41:21 F
Ext TP FTP	↓ ↑ FTP	● Deployed	May 13th, 2021 06:48:02 PM PDT

Lifecycle actions for transports are:

- Create a transport: Adds a definition for the transport. The pair of B2B integrations are automatically created and permanently linked to this transport.
- Receive schedule: Only available for the FTP transport. This action navigates you to the schedule configuration page where you can define the polling schedule for the B2B integration for receiving messages. You should add a schedule prior to deploying the transport. This enables the deploy action to also automatically start your schedule.
- Deploy a transport: Makes the transport visible for runtime processing and also activates the B2B integrations. This transport can now receive and send messages.
- Redeploy a transport: Applies configuration changes to the runtime on-the-fly without disrupting message processing (some types of configuration changes require undeployment and redeployment). See Apply Configuration Changes To Runtime.
- Undeploy a transport: Hides the transport from runtime processing and also deactivates the B2B integrations. This transport can no longer receive and send messages.
- Delete a transport: Removes the definition and also deletes the B2B integrations.

After a transport is deployed, the status column displayed in the user interface is an overall view of the status of the two integrations.

A failed status for a transport usually means one of these integrations failed to activate or was (unexpectedly) manually deactivated.

Apply Configuration Changes To Runtime

Configuration changes to a transport are not applied to runtime processing unless you execute an action. The following table lists the type of change and the action needed.

Type of Configuration Change	Actions to Take to Apply the Changes at Runtime
Common configuration settings in a transport	Redeploy the transport.
Protocol-specific configuration changes in a transport (that is, AS2 or FTP settings), including changes to the values of identifiers	Redeploy the transport.
Changes to the connection properties (other than username/password)	Undeploy and then deploy the transport.
Changes to the username/password credentials in the connection	No actions are needed. The transport automatically uses the updated credentials if authentication failure occurs at runtime (older, cached credentials are discarded). If you want to force the change, undeploy and then deploy the transport.
Changes to certificates in the Certificate management page	Undeploy and then deploy the transport.

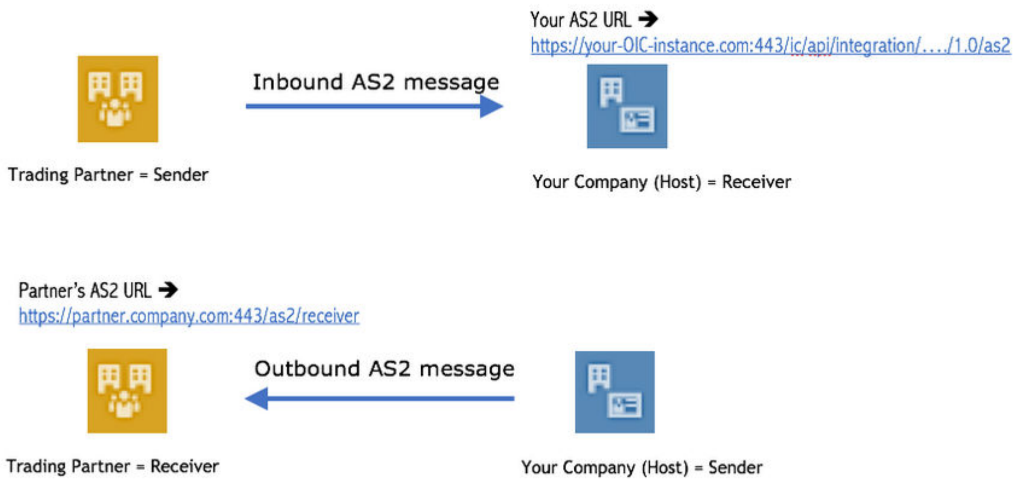
Collect AS2 Transport Details

You must collect AS2 transport details before you can define the AS2 transport in Oracle Integration.

- Collect AS2 Transport Details
- Credentials
- Certificates
- AS2 URL for Receiving

Collect AS2 Transport Details

AS2 is an HTTP-based, point-to-point protocol typically used for real-time transactions. A bidirectional AS2 message exchange involves two AS2 endpoints, as shown below:


Your AS2 URL ➔
https://your-OIC-instance.com:443/ic/api/integration/..../1.0/as2
Inbound AS2 message
Trading Partner = Sender
Your Company (Host) = Receiver
Partner's AS2 URL ➔
https://partner.company.com:443/as2/receiver
Outbound AS2 message
Trading Partner = Receiver
Your Company (Host) = Sender

While creating this transport, you must collect information from your trading partner about their AS2 endpoint. You also may need to provide information about your AS2 endpoint to your trading partner. The following table describes the information you need to collect:

For...	What You Need From Your Trading Partner	What You Need to Provide to Your Trading Partner
Basic connectivity	• The partner's AS2 URL. • An SSL certificate with a public key, if a self-signed certificate was used. • Username/password credentials for HTTP basic authentication, if enabled.	• Your AS2 URL (see AS2 URL for Receiving) • Username/password credentials (see Credentials)
Two-way SSL for outbound connections (an optional feature)	If you select the **Invoke** or **Trigger and Invoke** role, you can create a certificate alias to use for establishing client identity during two-way SSL communication.	See Prerequisites for Creating a Connection in *Using the AS2 Adapter with Oracle Integration Generation 2* .
Signed or encrypted AS2 messages (an optional feature)	• The partner's public certificate for signing and encryption. (Typically the same certificate is used for both signing and encryption, but if the partner prefers to use different ones, you should get two separate public certificates from them.)	• Your public certificate for signing and encryption (see Certificates)

Signing and encryption are optional features in AS2. You can start with only the basic connectivity first and add signing/encryption later. Signing/encryption provide nonrepudiation, message integrity, and security features and are recommended for production environments. However, there is a bit more complexity in setting those up.

The following table shows which PKI key is used in each scenario:

Message Configuration	Inbound Message	Outbound Message
Signed AS2 message	Partner's *public key for signing* is used to verify a signed message.	Your company's *private key for signing* is used to digitally sign the message.
Encrypted AS2 message	Your company's *private key for encryption* is used to decrypt the message.	Partner's *public key for encryption* is used to encrypt the message.

If you already have the AS2 endpoint information from your trading partner, follow these steps:

Step	Description
1	Upload each of the partner's certificates to **Home** > **Settings** > **Certificate**. Upload SSL certificates as X.509 Trust, whereas upload signing and encryption as X.509 Identity. For identity certificates, you decide and enter a unique alias. Note the aliases. See Manage Security Certificates in *Using Integrations in Oracle Integration Generation 2*.

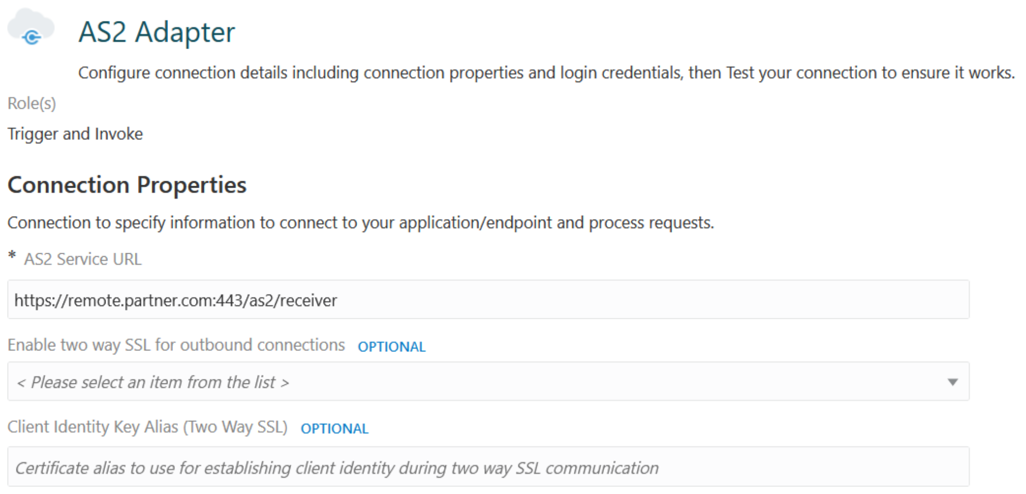
Step	Description
2	If signing/encryption is a requirement, acquire or generate a key-pair for signing and encryption (or two separate key-pairs, if you want to use separate keys for signing and encryption). See Certificates. Upload the private key to **Home** > **Settings** > **Certificate** as X.509 Identity and note the alias and password you enter. Share the public key with your trading partner. However, never share the private key.
3	Create an AS2 connection with the Trigger and Invoke role. In the Connections page, enter: • The partner's AS2 URL in the **AS2 Service URL** field • The username/password in the corresponding fields If signing/encryption are a requirement, configure the AS2 connection further. If both your partner and your company use one certificate for signing and encryption, select **AS2 Basic Policy**. If either of you use different certificates, select **AS2 Advanced Policy**. • For AS2 Basic Policy, enter the partner's certificate alias, corresponding to the identity certificate from step 1. • For AS2 Basic Policy, enter the private key alias and key password, corresponding to the identity certificate from step 2. • For AS2 Advanced Policy, enter each of the certificate aliases into the fields. See Configure Connection Security in *Using the AS2 Adapter with Oracle Integration Generation 2* .
4	Test the AS2 Adapter connection, to make sure it succeeds. If it fails, review the errors, verify that the AS2 URL entered is correct, and verify that the certificate aliases are correct. Save the AS2 Adapter connection.
5	Create an AS2 transport, selecting the AS2 Connection created in step 3. Complete the configuration. See Define an AS2 Transport.
6	Deploy the AS2 transport. After the state changes to deployed, the transport is ready for use.

If you do not yet have the AS2 endpoint information from your trading partner, but want to get your side ready for receiving AS2 messages, follow these steps:

Step	Description
1	Same as Step 1 in the previous table. Skip this step for now, but you can perform it when the information becomes available from the trading partner
2	Same as Step 2 in the previous table.
3	Same as Step 3 in the previous table, but given that the partner's AS2 URL is not yet available, enter a temporary placeholder URL in the **AS2 Service URL** field. This can be the URL of your Oracle Integration instance, copy and pasted from the browser URL address or any other valid URL. This placeholder is only needed to pass the connection test (which fails if the URL is invalid). Outbound AS2 messages do not work with this placeholder, but inbound messages can be received (since the AS2 service URL is not used when receiving inbound messages).
4	Same as Step 4 in the previous table.
5	Same as Step 5 in the previous table.
6	Same as Step 6 in the previous table.

An example of an AS2 Adapter connection is shown below.

- Connection properties:



AS2 Adapter

Configure connection details including connection properties and login credentials, then Test your connection to ensure it works.

Role(s)

Trigger and Invoke

Connection Properties

Connection to specify information to connect to your application/endpoint and process requests.

* AS2 Service URL

https://remote.partner.com:443/as2/receiver

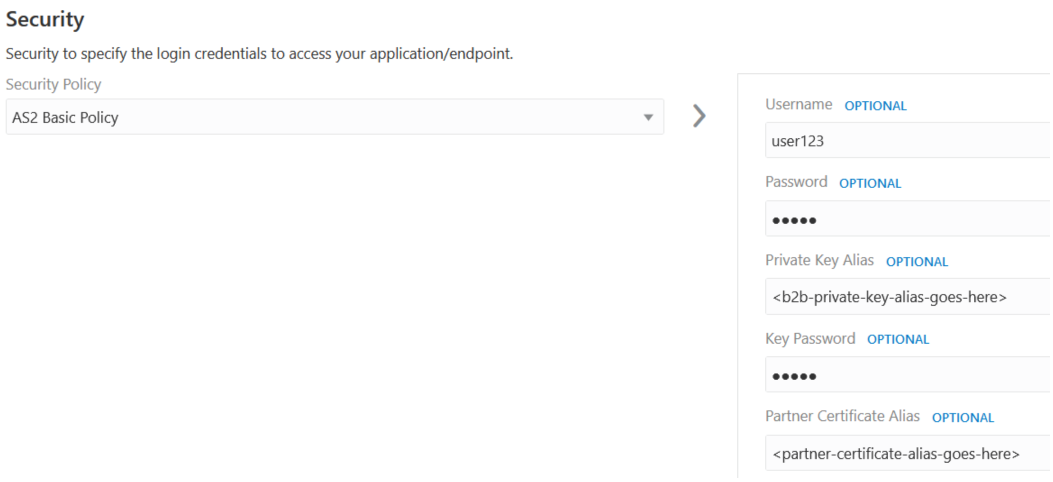
Enable two way SSL for outbound connections OPTIONAL

< Please select an item from the list >

Client Identity Key Alias (Two Way SSL) OPTIONAL

Certificate alias to use for establishing client identity during two way SSL communication

- Connection security:



Security

Security to specify the login credentials to access your application/endpoint.

Security Policy

AS2 Basic Policy

Username OPTIONAL

user123

Password OPTIONAL

•••••

Private Key Alias OPTIONAL

<b2b-private-key-alias-goes-here>

Key Password OPTIONAL

•••••

Partner Certificate Alias OPTIONAL

<partner-certificate-alias-goes-here>

Credentials

To receive messages over AS2 from an external trading partner, HTTP basic authentication is enforced. Your trading partner is required to send the Authorization HTTP header with username/password credentials you provide them in an AS2 message.

For internal testing you may use the same credentials that you use to log in to Oracle Integration to send test AS2 messages. However, it is not safe to share these credentials with an external trading partner because they can also log in to Oracle Integration with these credentials.

Instead, create a new user account in the Oracle Integration Identity Management application. Grant the Service Invoker role to this user account. This account is enough to send messages, but does not grant permissions to access any user interface pages in Oracle Integration. Share the username and password of this new user with the trading partner.

Certificates

If you want to enable encryption or signing for the AS2 communication, you must create a key pair and certificate following your company's process and generate a CA-signed certificate that you use for AS2 decryption and signing.

For testing with a self-signed certificate, here are simple steps to generate a key-pair using the Java `keytool`:

1. Generate public/private key pair using `keytool`.
   a. Specify any alias and a keystore file name, replacing `b2b-private-key-alias` and `b2b.jks` with your values.
   b. Enter a keystore password when prompted and note it.
   c. Enter your organization's information when prompted.

   This generates a key pair (a public key and associated private key) and self-signed digital certificate in a keystore. If the keystore does not exist, it is created.

   ```
   keytool -genkey -keyalg RSA -alias b2b-private-key-alias -validity
   1095 -keystore b2b.jks
   ```

2. Upload the JKS into Oracle Integration as the X.509 type (SSL transport) and Identity category using the same alias and password as entered above (this is part of Step 3 from the table of steps above).
3. Export the public key from this keystore as follows.
   a. Replace `b2b.jks`, `b2b-private-key-alias`, and *`public.cer`* with your keystore file name, alias that was used previously, and a file name to store the public certificate.

      ```
      keytool -export -keystore <b2b.jks> -alias <b2b-private-key-
      alias> -file <public.cer>
      ```

4. Convert it to any other industry-standard format using `keytool` as per your preference, if necessary. Share only the public certificate *`public.cer`* with your trading partner (never share the private key with anyone). Your trading partner uses the public key certificate for signature verification and encryption.

AS2 URL for Receiving

You need the AS2 URL for your AS2 endpoint to share with your trading partner. Once the transport is deployed (indicating it is ready to receive and/or send messages), your AS2 endpoint URL is displayed in the **AS2 Endpoint URL for Receiving** transport field. Copy this AS2 URL to share with your trading partner. This AS2 URL is not common across all trading partners; it is specific to the current trading partner that you are viewing or editing. Only that specific trading partner may send AS2 messages to this URL.

The AS2 URL is the URL to invoke the AS2 integration for receiving messages for this transport. While you can also get the same from the Integrations page, this provides an easier way to access it.

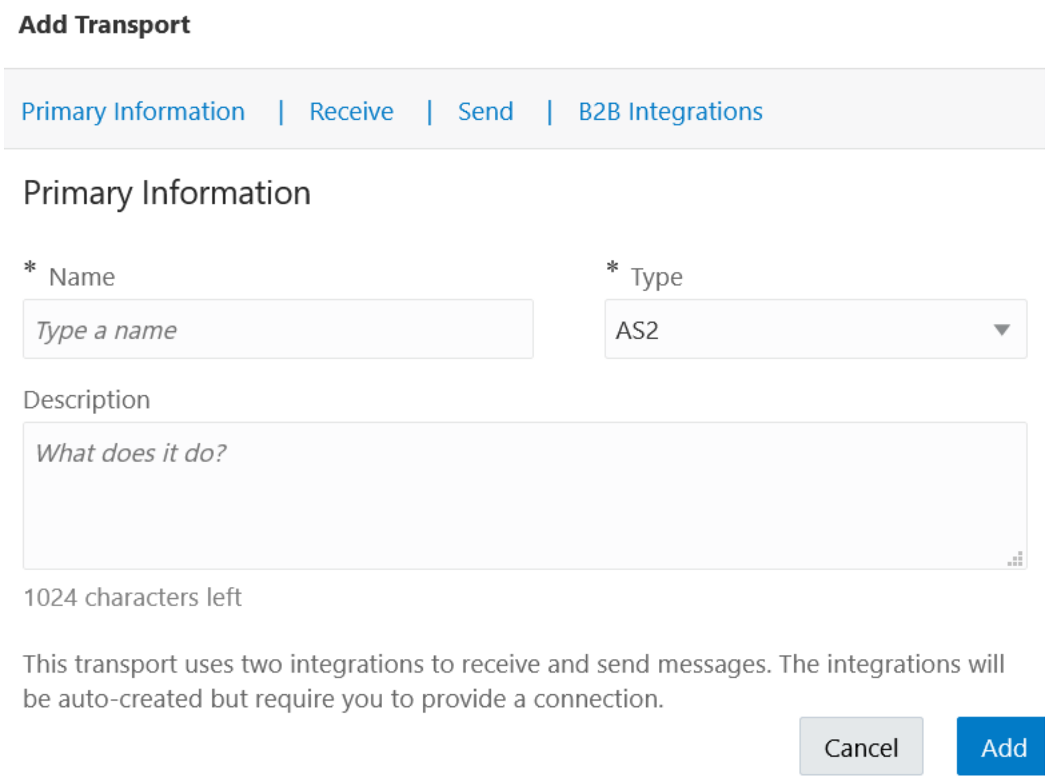
Define an AS2 Transport

Once you have collected AS2 transport details, you can define an AS2 transport in Oracle Integration.

1. In the left navigation pane, click **B2B** > **Trading Partners**.
   The Trading Partners page is displayed.
2. In the row of the trading partner for which to define transports, click ✎ .
3. Click **Transports & Agreements**.

Define Transports to Create Integrations

1. In the **Transports** section, click ⊕ to define how a message is delivered to or received from this trading partner. The configuration panel is opened:



Add Transport

Primary Information | Receive | Send | B2B Integrations

Primary Information

* Name: *Type a name*

* Type: AS2

Description: *What does it do?*

1024 characters left

This transport uses two integrations to receive and send messages. The integrations will be auto-created but require you to provide a connection.

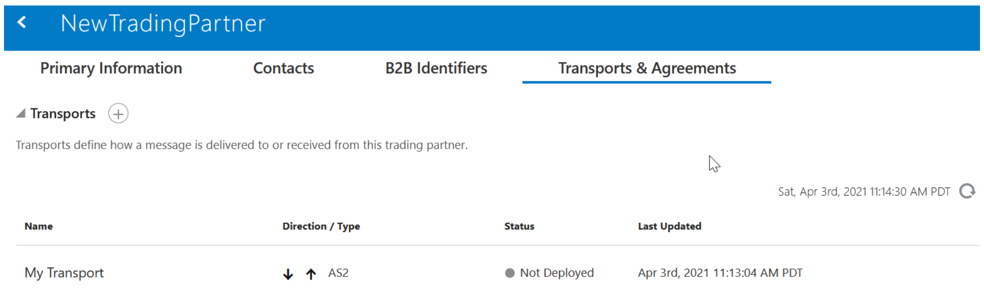
2. Define the following details.

Section	Description
Primary Information section • **Name**	Enter a name for the transport. The name is used for display purposes only.
• **Description**	Enter an optional description of the transport. The description is used for display purposes only.
• **Type**	Select **AS2** from the dropdown. This represents the communication protocol you use to exchange messages with your trading partner. Based on selecting AS2, the appropriate configuration settings for the AS2 transport are displayed.
• **Trading partner's connection (Trigger and Invoke)**	Select an existing AS2 Adapter connection configured for connectivity to your trading partner or click ⊕ to create a new AS2 Adapter connection on the Connections page. It must be a Trigger and Invoke connection. Trigger Only and Invoke Only connections cannot be used for transports. See Create a Connection in *Using the AS2 Adapter with Oracle Integration Generation 2* . If you want to select another connection, you can do so when this transport is not deployed. Once you deploy the transport, the connection selection cannot be changed. You can modify the configuration properties inside the connection at any time. However, if you modify the connection settings after this transport is deployed, you must undeploy and then redeploy the transport for the changes to take effect.
• **Partner Identifier**	Used as an `AS2-To` header for outbound messages and as the expected `AS2-From` header for inbound messages. See Define B2B Identifiers.
• **Host Identifier**	Used as the `AS2-From` header for outbound messages and as the expected `AS2-To` header for inbound messages. See Host Profile.
• **Character Encoding**	Select the character encoding to apply to all payloads processed through this transport. The character encoding is used at the EDI parsing (inbound) or EDI generation (outbound) step.
Receive section • **Allow any AS2 Identifier defined for this trading partner**	Enable this checkbox if you want to accept an `AS2-From` header value from within a set of possible values (as opposed to accepting just one value). A typical case for this is when different business units within the trading partner's organization use different AS2 identifiers, and you want to accept messages from all of them. For this to work, you must also add all acceptable `AS2-From` values as B2B identifiers to the trading partner. If this is disabled, only one specific AS2 identifier selected for **Partner's Identifier** is accepted at runtime for inbound processing.
• **Do not validate the AS2-To header**	Enable this if you want to allow any value for the `AS2-To` header, effectively disabling the strict validation otherwise done against the host's AS2 identifier.

Section	Description
• **AS2 Endpoint URL for Receiving**	This is a display-only field. After the transport is deployed, indicating it is ready to receive and send messages, your AS2 endpoint URL is displayed. You may share this URL with your trading partner. This AS2 URL is not common across all trading partners. It is specific to the current trading partner that you are viewing or editing.
Send section • **AS2 Subject**	An optional field that is inserted into all outbound messages that use this transport, as the `Subject` HTTP header.
• **Content-Type**	The payload's type for outbound messages. Typically for EDI payloads, use **application/EDI-Consent** as a generic way to specify that the payload can be either X12 or EDIFACT.
• **User-defined Content-Type**	To enter another value not already available in the **Content-Type** drop-down list, select **User-defined Type** and enter your value.
• **Signature**	If you want to enable message signing for outbound messages, select the appropriate signing algorithm from the drop-down list. For signing, you must configure the AS2 Adapter connection appropriately with certificates, as described in Step 3 of the above table. This field does not apply for inbound message processing. Signature verification of signed inbound messages is done automatically and there is no configurable option to enable or disable it. You must configure the AS2 connection with the right certificates for that to work also.
• **Encryption**	If you want to enable message encryption for outbound messages, select the appropriate encryption algorithm from the drop-down list. For encryption, you must configure the AS2 Adapter connection appropriately with certificates as described in Step 3 of the above table. This field does not apply for inbound message processing. Decryption of encrypted inbound messages is done automatically and there is no configurable option to enable or disable it. You need to configure the AS2 Adapter connection with the correct certificates for that to work also.
• **Compression**	Optionally select to compress the outbound message. • **None** • **Digitally sign first, then compress**: Sign the outbound message before compressing it. • **Compress first, then digitally sign**: Compress the outbound message before signing it.

Section	Description
• **Request MDN**	Select a value in the drop-down list to request a synchronous or asynchronous MDN (or **None** for no MDN) when sending outbound messages. This field does not apply to inbound message processing. MDN is automatically generated and sent back to the trading partner based on whether your trading partner has requested an MDN as part of an inbound message. There is no configuration required. The AS2 HTTP headers `Disposition-Notification-To`, `Disposition-Notification-Options`, and `Receipt-Delivery-Option` convey whether the partner wants to receive an MDN back, whether it should be synchronous or asynchronous, and whether it needs to be signed. The AS2 transport automatically handles the MDN processing. See the AS2 specification (RFC 4130) to understand more technical details. • **None**: Request that no MDN be sent back. • **Async MDN**: Request that the MDN be sent separately from the outbound message. • **Sync MDN**: Request that the MDN be sent immediately in the response.
• **Request Signed MDN**	If you choose to request an MDN for an outbound message, select the checkbox to ask the trading partner to send signed MDNs. You must configure the AS2 Adapter connection appropriately with certificates as described in Step 2 of the table of steps above, so that the signed MDNs can be validated.
B2B Integrations section • **Integration name prefix**	Enter a short prefix that is used to form the complete integration names for receiving messages and sending messages. For the AS2 transport, it forms the integration names: ***your_prefix* AS2 Receive** and ***your_prefix* AS2 Send**. Details about these integrations are provided. See Create B2B Integrations for Receiving and Sending.
• **Integration identifier prefix**	Enter a short prefix that is used to form complete integration identifiers for receiving messages and sending messages. For the AS2 transport, it forms the integration identifiers: ***your_prefix*_AS2_RECEIVE** and ***your_prefix*_AS2_SEND**. The final integration identifier must be unique across all integrations. Therefore, ensure that you enter a prefix that is unique. If the uniqueness check fails, you get the opportunity to try with a different prefix.

3. Click **Add**.
   The new transport is displayed.


NewTradingPartner

Primary Information | Contacts | B2B Identifiers | Transports & Agreements

Transports

Transports define how a message is delivered to or received from this trading partner.

Sat, Apr 3rd, 2021 11:14:30 AM PDT

Name	Direction / Type	Status	Last Updated
My Transport	↓ ↑ AS2	● Not Deployed	Apr 3rd, 2021 11:13:04 AM PDT

4. From the ☰ menu, select **Deploy**.
5. Select **Deploy** again when prompted.
   If successful, the following message is displayed.

   ```
   Transport transport_name was deployed successfully.
   ```

   The transport status is changed to **Active**.
6. Go to the Integrations page and note that both integrations are created and activated.
7. If you need to undeploy the transport, select **Undeploy** from the ☰ menu in the **Transports & Agreements** section. Undeploying the transport also undeploys the integrations.

Collect FTP Transport Details

FTP is a file-based transfer protocol that requires an external FTP server. The FTP transport only acts as an FTP client that connects directly to your trading partner's FTP server or an FTP server hosted by your company (which the trading partner also connects to as a client). In either case, the FTP transport pulls files from (or pushes files to) an FTP server.

An FTP transport works on a time-based schedule when receiving files (as file polling), and in real-time when sending files. FTP is typically used in a batch mode (for example, processing a batch of purchase orders or a sales catalog on a nightly schedule).

Before creating this transport, you need to create an FTP Adapter connection with the Trigger and Invoke role. Review all prerequisites and details to create an FTP Adapter connection. See Create an FTP Adapter Connection in *Using the FTP Adapter with Oracle Integration Generation 2*.

Note:

Note these FTP transport restrictions (although these features are provided in the FTP Adapter in standalone mode):

- Connectivity to an on-premises FTP server through the connectivity agent is not currently supported.
- PGP encryption and decryption are not currently supported.
- Signing and signature verification for files is not currently supported.
- Processing of `.zip` files is not currently supported.

Define an FTP Transport

Once you have collected FTP transport details, you can define an FTP transport in Oracle Integration.

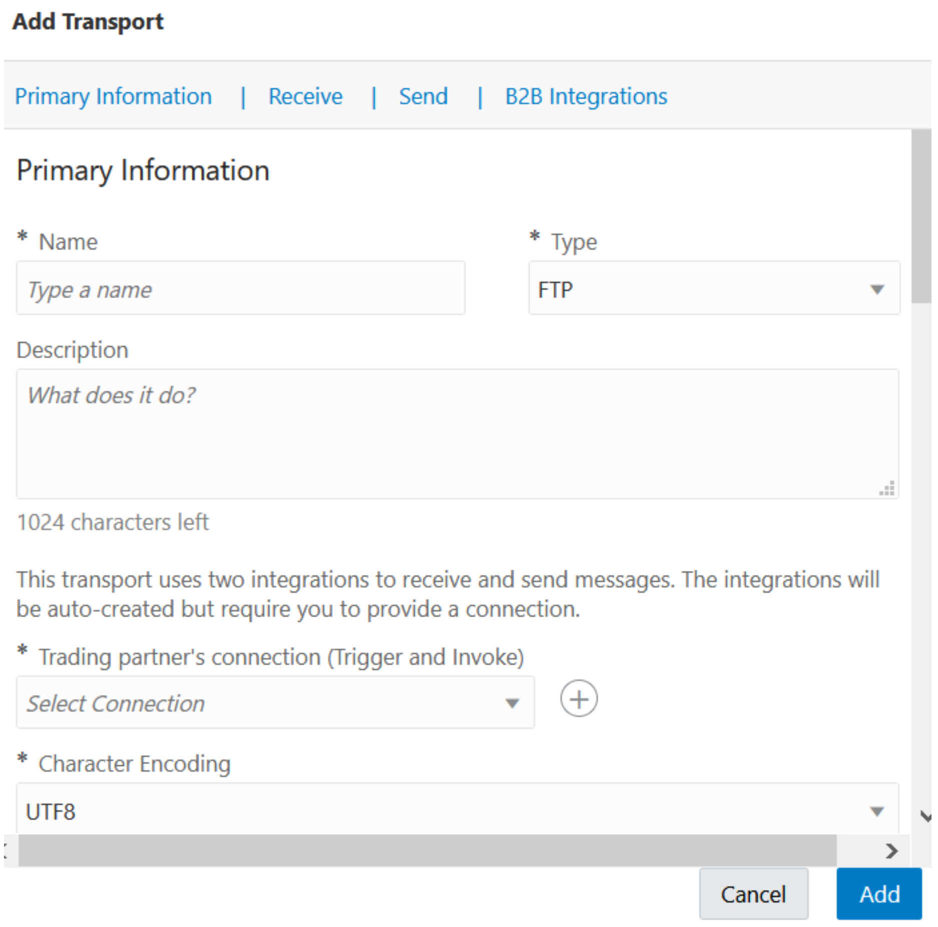
1. In the left navigation pane, click **B2B** > **Trading Partners**.

The Trading Partners page is displayed.

2. In the row of the trading partner for which to define transports, click

   .
3. Click **Transports & Agreements**.
4. In the **Transports** section, click

   to define how a message is delivered to or received from this trading partner. The configuration panel is opened:


Add Transport

Primary Information

* Name: Type a name

* Type: FTP

Description: What does it do?

1024 characters left

This transport uses two integrations to receive and send messages. The integrations will be auto-created but require you to provide a connection.

* Trading partner's connection (Trigger and Invoke): Select Connection

* Character Encoding: UTF8

5. Define the following details.

Section	Description
Primary Information section • **Name**	Enter a name for the transport. The name is used for display purposes only.
• **Description**	Enter a description of the transport. The description is used for display purposes only.

Section	Description
• **Type**	Select **FTP** from the drop-down list. This represents the communication protocol you use to exchange messages with your trading partner. Based on selecting FTP, the appropriate configuration settings for the FTP transport are displayed.
• **Trading partner's connection (Trigger and Invoke)**	Select an existing FTP Adapter connection to use or click ⊕ to create a new FTP Adapter connection on the Connections page. It must be a Trigger and Invoke connection. Trigger Only and Invoke Only connections cannot be used for transports. See Create a Connection in *Using the FTP Adapter with Oracle Integration Generation 2*. If you want to select another connection, you can do so when this transport is not deployed. Once you deploy the transport, the connection selection cannot be changed. You can modify the configuration properties inside the connection at any time. However, if you modify the connection settings after this transport is deployed, you must undeploy and then redeploy the transport for the changes to take effect.
• **Character Encoding**	Select the character encoding to apply to all payloads processed through this transport. The character encoding is used at the EDI parsing (inbound) or EDI generation (outbound) step.
Receive section • **Input Directory**	Specify a directory path on the FTP server to poll for files (inbound). For example: `/b2b/inbound` If the input directory is specified, the FTP transport is considered capable of receiving, and the direction indicates a darker up-arrow in the transports listing.
• **Scan Recursively**	Select to scan all subdirectories to look for files. Deselect to only scan the immediate input directory, ignoring any subdirectories.
File Name Filter	Enter a wildcard expression that the FTP server understands to match files. For example, `*.edi`.

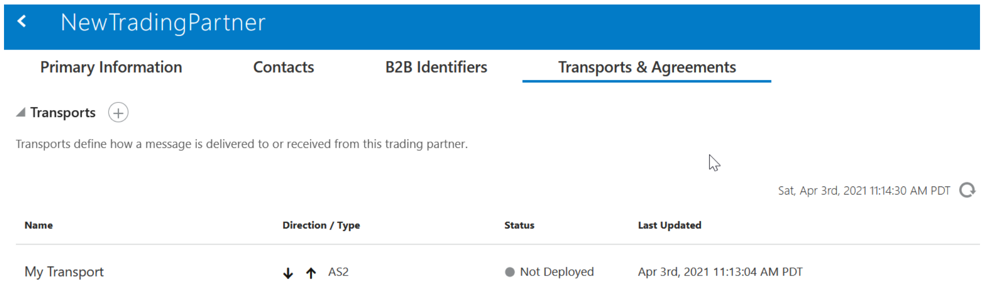
Section	Description
• **Minimum Age (Seconds)**	Specify a delay in seconds. The value specified indicates how long to ignore newly created files, relative to their creation time. For example, if a file is created at `11:02:00` and a minimum age of `60` seconds is specified, that file is ignored and not picked up for processing until `11:03:00`, when it becomes `60` seconds old. This delay allows the writer of a file to complete its transfer of bytes and avoid situations where not enough time was given to complete the file transfer. As a result, a half-written file was picked up for processing. If you specify a nonzero minimum age, make sure to also select an appropriate value for the FTP server time zone drop-down in the FTP Connections page. The time zone calculates the current age of a file, based on the current time and the creation timestamp of the file. If you do not select a time zone, then it defaults to the time zone of the Oracle Integration server. The mismatch can delay the processing of files for up to 12 hours. See Configure Connection Properties in *Using the FTP Adapter with Oracle Integration Generation 2*.
• **Max Count of Files**	Specify the maximum number of files to be processed in one scheduled call. If the schedule is every hour, then every hour up to the maximum number of files are processed and any remaining files are picked up in a future run. See Create Scheduled Integrations in *Using Integrations in Oracle Integration Generation 2*. Limit this to a reasonable number so that the processing integration does not run for a very long time and consume precious resources just for one trading partner. The maximum value is 1000. The default is 100. **Note**: The file list is returned in a sorted order according to the last modified time. If you selected 10 as the maximum number of files and the last modified time of the eleventh file is the same as the tenth file, then the eleventh file is also added. This continues until you get a file with a different timestamp.
Send section • **Output Directory**	Specify the directory path on the FTP server in which to put outbound files. For example: `/b2b/outbound` If the output directory is specified, the FTP transport is considered capable of sending, and the direction indicates a darker down-arrow in the transports listing.

Section	Description
• **Output File Name**	Specify an output file name. The file name can include substitution patterns, for example, (`Out%SEQ%.edi` creates files with names like: `Out1.edi`, `Out2.edi`, and so on). The following patterns are supported (surround them inside `%` chars): • `SEQ` • `yyyyMMdd` • `MMddyyyy` • `yyMMddHHmmss` • `yyMMddHHmmssSS` • `yyMMddHHmmssz` • `yyMMddHHmmssSSz` **Note**: Use `%SEQ%` with caution because concurrent processing of messages may generate duplicate sequence numbers in some cases. This causes files to be overwritten.
B2B Integrations section • **Integration name prefix**	Enter a short prefix that is used to form the complete integration names for receiving messages and sending messages. For the FTP transport, it forms the integration names: ***your_prefix*** **FTP Receive** and ***your_prefix*** **FTP Send**. Details about these integrations are provided. See Create B2B Integrations for Receiving and Sending.
• **Integration identifier prefix**	Enter a short prefix that is used to form complete integration identifiers for receiving messages and sending messages. For the AS2 transport, it forms the integration identifiers: ***your_prefix*_FTP_RECEIVE** and ***your_prefix*_FTP_SEND**. The final integration identifier must be unique across all integrations. Therefore, ensure that you enter a prefix that is unique. If the uniqueness check fails, you get the opportunity to try with a different prefix.

Note:

There is one behavior not exposed in the transport configuration for FTP. It controls what should happen to an inbound file after processing is complete. You can alter this behavior by changing a property for the FTP receive integration. See FTP Receive Integration Postprocessing Behavior.

6. Click **Add**.
   The new transport is displayed.



Transports define how a message is delivered to or received from this trading partner.

Sat, Apr 3rd, 2021 11:14:30 AM PDT

Name	Direction / Type	Status	Last Updated
My Transport	↓ ↑ AS2	● Not Deployed	Apr 3rd, 2021 11:13:04 AM PDT

7. If you selected FTP as the transport protocol, select **Receive Schedule** from the menu to define a schedule. Once the schedule is created and saved, click to return to the **Transports & Agreements** section. See Define the Integration Schedule in *Using Integrations in Oracle Integration Generation 2*.
8. From the menu, select **Deploy**.
9. Select **Deploy** again when prompted.
   If successful, the following message is displayed.

   ```
   Transport transport_name was deployed successfully.
   ```

   The transport status is changed to **Active**.
10. Go to the Integrations page and note that both integrations are created and activated.
11. If you need to undeploy the transport, select **Undeploy** from the menu in the **Transports & Agreements** section. Undeploying the transport also undeploys the integrations.

FTP Receive Integration Postprocessing Behavior

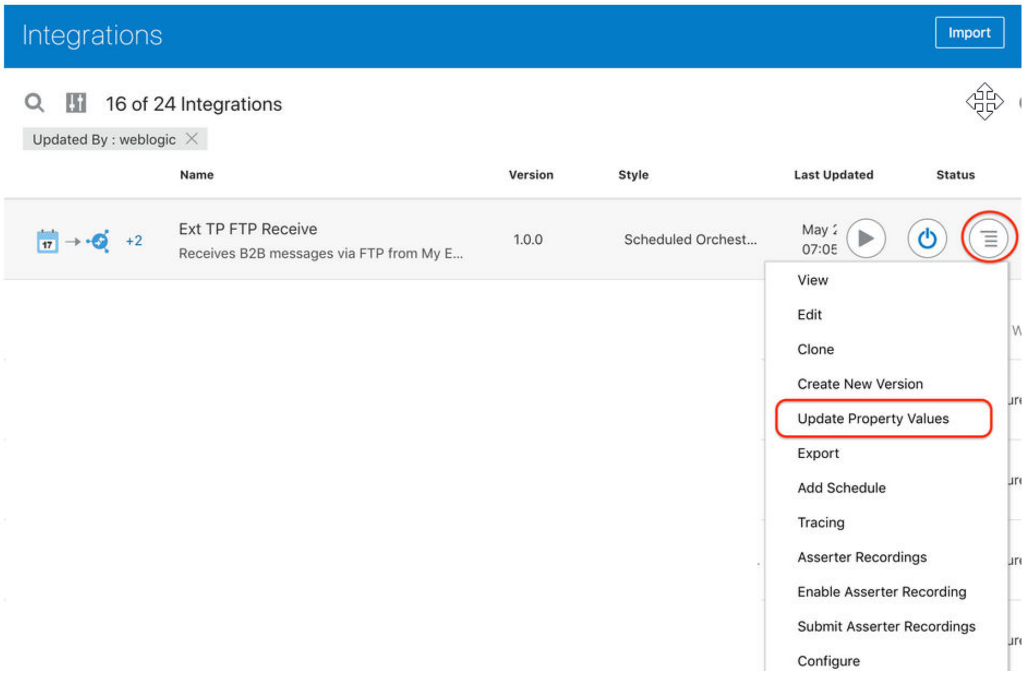
By default, files that are processed by the FTP Receive integration are moved to a backup directory. The backup directory name is assumed to be `original-input-directory_backup` and is created automatically provided the correct permissions are granted to create it.

You can customize this behavior as follows. To locate these properties, use the **Update Property Values** menu as described below.

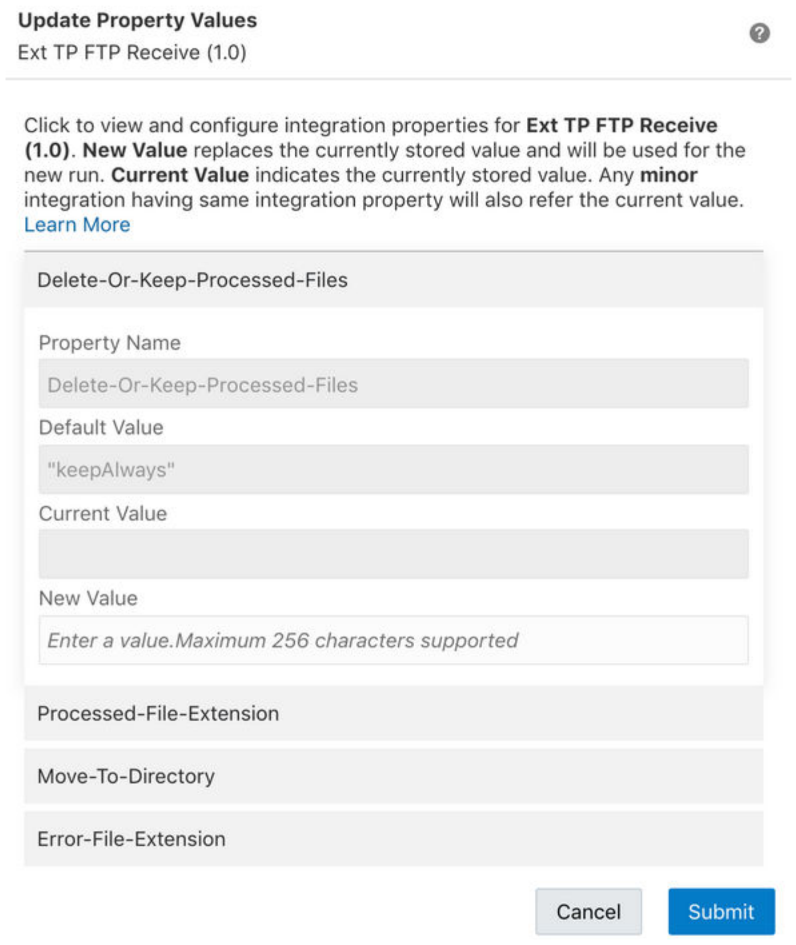
Permission	Description	Values
`Delete-Or-Keep-Processed-Files`	Controls whether processed files are retained or deleted.	• `deleteAlways`: Deletes a processed file regardless of whether processing succeeded or failed. • `keepOnError`: Deletes a processed file only on success. If the processed file failed, the file is retained and moved to a different directory. • `keepAlways`: (default) Retains a processed file, regardless of whether processing succeeded or failed by moving it to a different directory.
`Error-File-Extension`	If a processing error occurs, rename the file by appending the *`Error-File-Extension`* value (for example, `_error`), depending on the `Delete-Or-Keep-Processed-Files` setting. For example, if an input file name was `850_po.edi`, then the default value after processing the file is renamed to `850_po.edi_error`.	• `_error` (default). • You can set the value for `Error-File-Extension` to any string. However, the combination of *`original-input-filenameError-File-Extension`* must result in a valid directory path on the FTP server.
`Move-To-Directory`	Processed files are moved to another directory depending on the `Delete-Or-Keep-Processed-Files` value. The backup directory is assumed to be *`original-input-directoryMove-To-Directory`*. For example, if the input directory was `/b2b/inbound`, then with the default value, processed files are moved to the `/b2b/inbound_backup` directory.	• `_backup` (default). • You can set the value for `Move-To-Directory` to any string. However, the combination of *`original-input-directoryMove-To-Directory`* must result in a valid directory path on the FTP server.

Permission	Description	Values
`Processed-File-Extension`	If processing is successful, rename the file by appending the *`Processed-File-Extension`* value (for example, `_processed`), depending on the `Delete-Or-Keep-Processed-Files` setting. For example, if an input file name was `850_po.edi`, then with the default value, after processing, the file is renamed to `850_po.edi_processed`.	• `_processed` (default). • You can set the value for `Error-File-Extension` to any string. However, the combination of *`original-input-filenameError-File-Extension`* must result in a valid directory path on the FTP server.

To change the behavior for one specific FTP transport, locate its FTP Receive Integration, and navigate to the Update Property Values dialog.



1. In the Update Property Values dialog, select the property to change and enter a value in the **New Value** field.
2. Select **Submit** to make the new values effective for future runs.



Update Property Values
Ext TP FTP Receive (1.0)

Click to view and configure integration properties for **Ext TP FTP Receive (1.0)**. **New Value** replaces the currently stored value and will be used for the new run. **Current Value** indicates the currently stored value. Any **minor** integration having same integration property will also refer the current value.
Learn More

Delete-Or-Keep-Processed-Files

Property Name
Delete-Or-Keep-Processed-Files
Default Value
"keepAlways"
Current Value

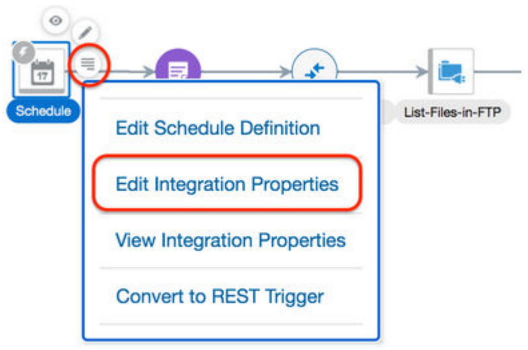
New Value
Enter a value.Maximum 256 characters supported

Processed-File-Extension

Move-To-Directory

Error-File-Extension

3. To change the behavior for all transports created in the future, edit the integration named **B2B Integration Template FTP Receive**.
4. In the schedule action in the integration canvas, select **Edit Integration Properties** and change the default values.



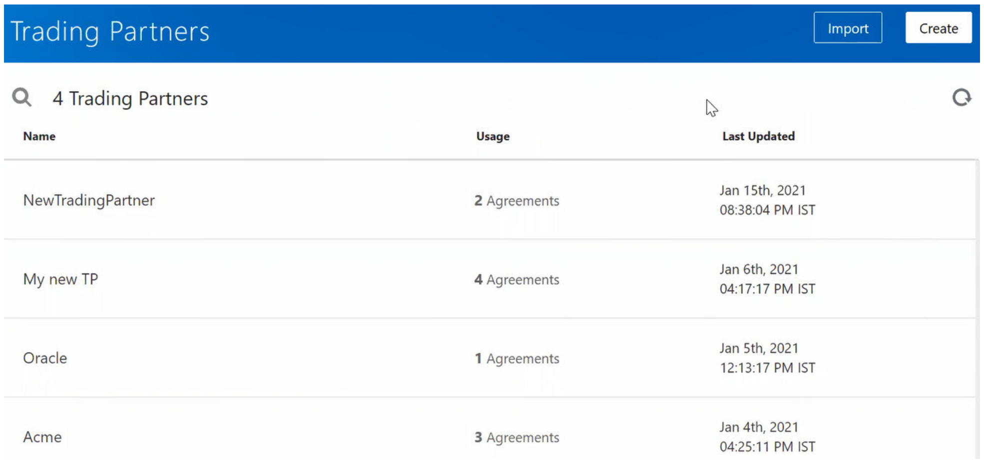
Create Agreements

This chapter providing details about creating and managing agreements. You define one or more agreements for a B2B trading partner with an intent to send or receive only certain types of business documents to or from that trading partner.

Detailed agreement concepts are provided. See Agreements.

You can view a list of inbound agreements and outbound agreements from the trading partner's **Transports & Agreements** tab.

1. In the left navigation pane, click **B2B** > **Trading Partners**.
   The Trading Partners page is displayed.

Name	Usage	Last Updated
NewTradingPartner	2 Agreements	Jan 15th, 2021 08:38:04 PM IST
My new TP	4 Agreements	Jan 6th, 2021 04:17:17 PM IST
Oracle	1 Agreements	Jan 5th, 2021 12:13:17 PM IST
Acme	3 Agreements	Jan 4th, 2021 04:25:11 PM IST

   Note the following details:

   - You cannot delete a trading partner with active agreements.
   - Click 👁 to view specific details about the trading partner, including any associated trading partner agreements.

2. In the row of the trading partner for which to create agreements, click ✎ .

Define Inbound and Outbound Agreements

1. Click **Transports & Agreements**.
2. In the **Inbound Agreements** section, click ⊕ to add a new agreement.
   a. Define the following details.

Field	Description
Name	Enter a name. This is only used for your reference. Your trading partner does not see any of the agreement names or configuration details you define.
Description	Enter an optional description.
Select a Document	Select the type of document to receive. You can select an existing B2B document or create a new B2B document.
Select a backend integration	Select a backend integration to which to route this document after B2B processing. Either select one from the list or, if you know the identifier and version of your backend integration, enter it in the following format: `INTEGRATION_IDENTIFIER|01.00.0000` Backend integration concepts are provided. See Integrations Used for B2B Message Processing. Ensure that you have created a backend integration. See Create Backend Integrations. The dropdown list of integrations is filtered and only shows integrations that use the B2B action. It also currently shows all B2B system integrations. Do *not* select a B2B system integration. You must select a backend integration that you created separately to handle this type of document. A backend integration interfaces with your backend application, such as Oracle ERP Cloud. **Note**: Integrations specific to B2B for Oracle Integration can be displayed by entering `ediadapter` in the **Search** field on the Integrations page.
Enable Validations	Select to perform syntactical validations on the received EDI payload and reject it if validation errors are found. If **Generate Functional Ack** is also enabled, the functional acknowledgment sent back to the trading partner conveys the acceptance or rejection, including any validation errors found. Deselect to skip syntactical validation checks and always accept the EDI payload.

Field	Description
Generate Functional Ack	Select to generate and send a functional acknowledgment back to your trading partner. • For EDI X12, a 997 document is generated. • For EDIFACT, a CONTRL document is generated to relay the outcome of the EDI translation. The generated functional acknowledgment is automatically routed to the B2B Integration for sending messages that is linked to the transport from which the incoming document was received. Deselect to not generate or send functional acknowledgments. This setting requires your company and trading partner to mutually decide up front whether or not to use functional acknowledgments. A problem scenario is when one party expects these acknowledgments, but the other party does not send them.
Enable Checks for Duplicate Control Numbers	Select to check for duplicate control numbers in B2B transactions in inbound agreements. This prevents processing of transactions with duplicate control numbers. For example, if duplication exists between the control numbers in two transactions, the first message is successfully processed because the number is unique, but the second transaction is caught and not processed. Transaction failure is visible in the **Message Logs** section of the transaction on the Track B2B Messages page. For example, here is part of the message for a duplicate control number that was caught and not processed: `This EDI transaction with document type [850], version [4010] was not translated because it was determined to be a duplicate of a transaction.`

b. Click **Add**.

c. From the

menu, select **Deploy**.

d. Select **Deploy** again when prompted.
The inbound agreement status is changed to **Active**.

e. If you need to undeploy the agreement, select **Undeploy** from the

menu.

f. Click

to make any updates.

3. In the **Outbound Agreements** section, click

to add a new agreement.

a. Define the following details.

Field	Description
Name	Enter a name. This is only used for your reference. Your trading partner does not see any of the agreement names or configuration details you define.
Description	Enter an optional description.
Select a Document	Select the type of document to send. You can select an existing B2B document or create a new B2B document.
Select identifiers	Certain mandatory and optional identifiers, such as EDI Interchange ID, are inserted into EDI envelope segments during the outbound translation. Select the identifiers you want inserted into the EDI envelopes. See Define B2B Identifiers and Define Identifiers in the Host Profile. • **Select Trading Partner Identifiers**: Identifies the trading partner receiving the document. Select trading partner identifiers that you want to insert as the receiver-side values (for example, Interchange Receiver ID). • **Select Host Identifiers**: Identifies the host sending the document. Select host identifiers that you want to insert as the sender-side values (for example, Interchange Sender ID).
Select a Transport	Select a transport to route messages processed through the current outbound agreement to that transport for final delivery to the external trading partner. Selecting a transport defines a routing rule that controls how a message gets routed from a backend integration to a B2B integration for sending messages. Details about how outbound messages are routed are provided. See Message Routing Between Integrations.

Field	Description
Enable Validations	• Select to perform syntactical validations on the canonical XML payload during the outbound EDI translation. If validation errors are detected at this step, it means that the output EDI is syntactically invalid and must not be sent to the trading partner. Therefore, it is rejected during the translation phase and not routed to the B2B integration for sending messages. • Deselect to skip syntactical validation checks and send the EDI payload to the trading partner even if it has errors.
Functional Ack Required	• Select to require (and expect) that your trading partner always send back functional acknowledgments. When an acknowledgment is required, an outbound business message goes into a **Pending Functional Ack** state after transmission and is marked as **Successful** or **Failed** only when an acknowledgments is received. • Deselect to not require (or expect) functional acknowledgments. Outbound business messages are immediately marked as **Successful**. This setting requires your company and your trading partner to mutually decide up front whether or not to use functional acknowledgments. A problem scenario is when one party expects these acknowledgments, but the other party does not send them.

b. Click **Add**.
c. From the

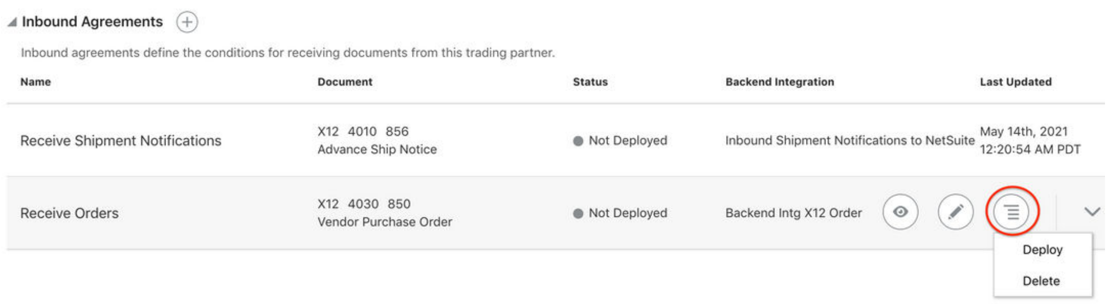
   menu, select **Deploy**.
d. Select **Deploy** again when prompted. The outbound agreement status is changed to **Active**.
e. If you need to undeploy the agreement, select **Undeploy** from the

   menu.
f. Click

   to make any updates.

Lifecycle Actions for Agreements

Click on the action menu on a row to view available actions.



Inbound Agreements

Inbound agreements define the conditions for receiving documents from this trading partner.

Name	Document	Status	Backend Integration	Last Updated
Receive Shipment Notifications	X12 4010 856 Advance Ship Notice	● Not Deployed	Inbound Shipment Notifications to NetSuite	May 14th, 2021 12:20:54 AM PDT
Receive Orders	X12 4030 850 Vendor Purchase Order	● Not Deployed	Backend Intg X12 Order	

Lifecycle actions for agreements are:

- Create an agreement: Adds the definition in design-time only (but unless deployed, the new agreement is not enforced at runtime).
- Deploy an agreement: Makes the agreement visible for runtime processing and is immediately enforced.
- Redeploy an agreement: Applies configuration changes to the runtime on-the-fly without disrupting message processing.
- Undeploy an agreement: Hides the agreement from runtime processing, making it no longer effective, starting immediately.
- Delete an agreement: Removes it from design-time.

Updating Values and Applying Changes To Runtime

You can update existing agreement settings at any time. However, the changes are not applied to the runtime processing until the agreement is redeployed.

Redeployment is a lifecycle action available for agreements. It applies changes to the runtime, on-the-fly, without disruption to message processing.

Changes to Identifier Values

Even if you don't directly change the agreement's settings, you cause an indirect change to an agreement if you modify values for an identifier from **Trading Partner** > **B2B Identifiers** or **Host Profile** > **Identifiers**.

To apply indirect changes, redeploy the agreement.

There is one special case for inbound agreements, because B2B identifiers are implicitly used by inbound agreements:

- If you add, update, or delete trading partner's identifiers, redeploy any of the agreements to apply the identifier changes to runtime.

Export and Import a Trading Partner

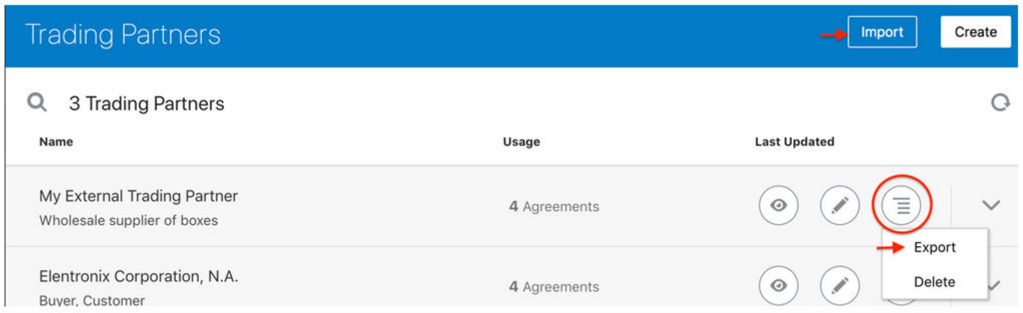
You can export an individual trading partner and import it into another instance of Oracle Integration. Importing a trading partner also imports the associated agreement references (for example, documents, schemas, and identifier references).

Note:

Ensure that you also separately import any backend integrations referenced in the trading partner agreement.

Export a Trading Partner

You can export or import a trading partner definition from the trading partners listing page.



Name	Usage	Last Updated
My External Trading Partner Wholesale supplier of boxes	4 Agreements	
Elentronix Corporation, N.A. Buyer, Customer	4 Agreements	

1. In the left navigation pane, click **B2B** > **Trading Partners**.
2. From the

   menu, select **Export**.
3. Download the ZIP file.

Import a Trading Partner

1. In the left navigation pane, click **B2B** > **Trading Partners**.
2. Click **Import**.
3. If you want to overwrite an existing trading partner agreement of the same name, click **Overwrite Any Existing Drafts**.
   This selection overwrites the trading partner and all associated artifacts (identifiers and agreements) when there are no active agreements. Associated documents and schemas are also overwritten if they do not have any active agreement references for other trading partners.

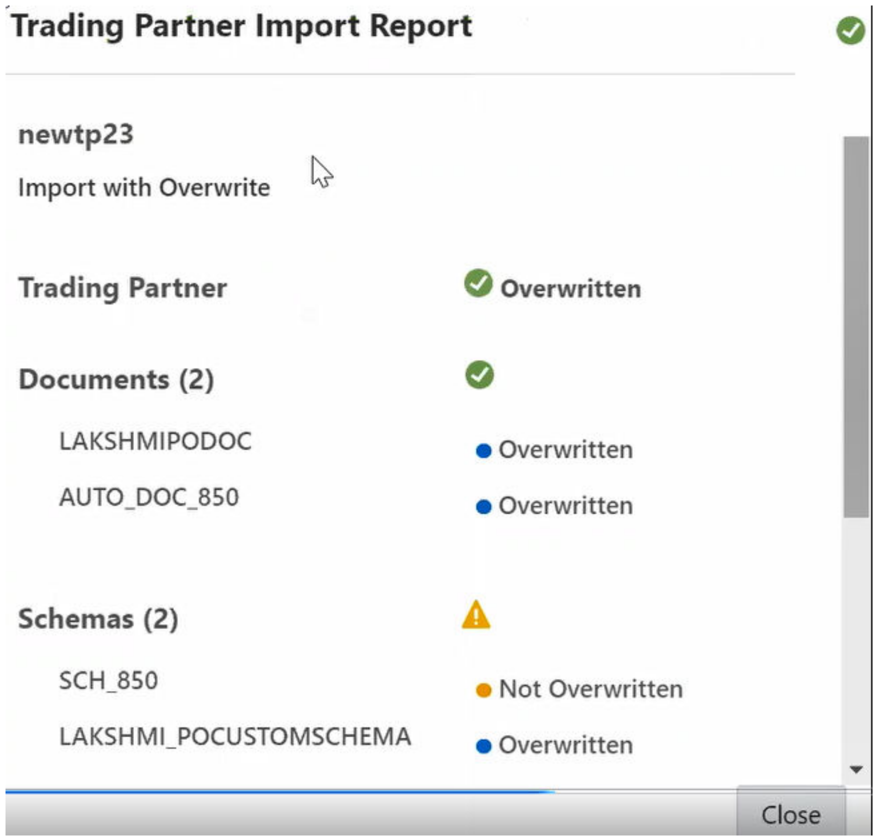
   If you select to overwrite, you must click the checkbox explicitly for the import to overwrite the existing trading partner, associated documents, and schemas. If the checkbox is not selected, the artifact import is skipped (that is, the document and schemas are skipped during the import if they already exist in the system).

If these documents and schemas have other references (that is, if another agreement for a different trading partner refers to the same document or schema), then regardless of whether you selected to overwrite, the import action is skipped for the document and schemas. Artifacts can be overwritten only if they do not have any other dependencies. For example, a schema cannot be overwritten because it is referenced by another B2B document used by another trading partner.

4. If import is successful, a message is displayed:

   `B2B Trading Partner import successfully for` *`trading_partner_name`*`.` **`More Information.`**

5. Click **More Information** to view the Trading Partner Import Report, which provides details about what was overwritten and not overwritten.



Trading Partner Import Report

newtp23

Import with Overwrite

Trading Partner	✓ Overwritten
Documents (2)	✓
LAKSHMIPODOC	● Overwritten
AUTO_DOC_850	● Overwritten
Schemas (2)	⚠
SCH_850	● Not Overwritten
LAKSHMI_POCUSTOMSCHEMA	● Overwritten

6. Click **Close**.
7. Click ✎, then click the **Agreements** tab.
8. Verify that all references are intact (for example, if any backend integrations require importing and reactivating) and make any necessary updates.
9. Redeploy the agreements.

Create B2B Integrations for Receiving and Sending

When you add a transport to a trading partner, two integrations are automatically created for you: the B2B integration for receiving and the B2B integration for sending (these are not their exact names, but rather indicative of the concept).

Both integrations are front-end integrations because they directly interface with the external trading partner. These integrations are at the heart of the transport for its runtime functioning.

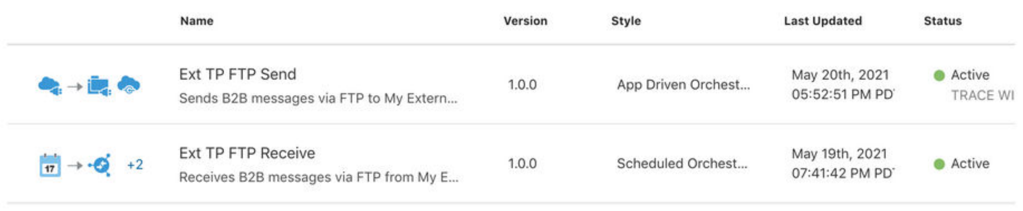
These integrations are automatically created when you define a transport. Two master template integrations are cloned internally to create these two integrations for the transport. The names of these master template integrations are as follows:

- B2B Integration template FTP Receive and B2B Integration template FTP Send: Used as the master template for the FTP transport.
- B2B Integration template AS2 Receive and B2B Integration template AS2 Send: Used as the master template for the AS2 transport.

The master template integrations are included with B2B for Oracle Integration. They are not initially present in your Oracle Integration instance, but created on-demand when you create your first transport.

When you create a transport, you have some control over the actual names of these B2B integrations. In the transport settings, you are prompted for an integration name prefix and an integration identifier prefix. These prefixes are used to generate the actual names and identifiers for the B2B integrations.

For example, the automatically created B2B integrations look as follows when the name prefix entered is `Ext TP`.



	Name	Version	Style	Last Updated	Status
	Ext TP FTP Send Sends B2B messages via FTP to My Extern...	1.0.0	App Driven Orchest...	May 20th, 2021 05:52:51 PM PD	● Active TRACE WI
+2	Ext TP FTP Receive Receives B2B messages via FTP from My E...	1.0.0	Scheduled Orchest...	May 19th, 2021 07:41:42 PM PD	● Active

In general, you don't need to edit or manage the B2B integrations directly because most functionality is driven through lifecycle actions on the transport.

Note:

You may customize the B2B integrations. However, in future versions, Oracle may provide enhancements to the master template integrations and a way to upgrade a transport, which overwrite the B2B integrations so as to leverage new features. You may have to add any customizations again into the integrations.

There are a few cases where you may need to do some management directly on the B2B integrations, specifically for the FTP transport.

1. View Past and Future Scheduled Runs for an FTP Receive Integration - refer to various other functions you can perform. See Create Scheduled Integrations in *Using Integrations in Oracle Integration Generation 2*.
2. Use submit now to trigger ad-hoc runs and make the FTP Receive integration execute immediately to poll for files.
3. Change the default behavior for what should happen to files, in case of FTP Receive, after the inbound processing is completed. See FTP Receive Integration Postprocessing Behavior.

Create Backend Integrations

You create one backend integration for each unique document definition that you want to receive or send. There are two types of backend integrations, depending on the direction:

- Inbound backend integration: Handles documents received from a partner.
- Outbound backend integration: Handles documents sent to a partner.

This section explains how many of these integrations you need to create. Learn the concepts behind backend integrations. See Integrations Used for B2B Message Processing.

Let's take an example scenario:

Document Types	How Many Backend Integrations You Need
You are expecting to receive the following types of inbound documents from various trading partners: • Purchase order acknowledgments • Invoices • Advanced shipment notices	You need three inbound backend integrations. (You can collapse the three into a single backend integration, if that is your design preference.)
You are expecting to send the following types of outbound documents to various trading partners: • Purchase orders • Inventory inquiry	You need two outbound backend integrations. (You can collapse the two into a single backend integration, if that is your design preference.)

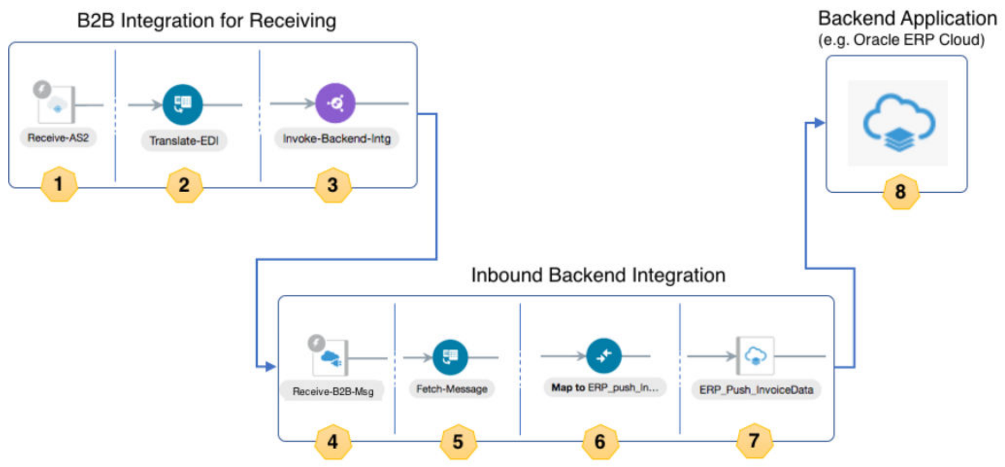
Let's take another variant of the first scenario:

Document Types	How Many Backend Integrations Do You Need
You are expecting to receive the following types of inbound documents from various trading partners: • Purchase order acknowledgments (uses the standard X12 schema) • Invoices (uses the standard X12 schema) • Invoices (uses a customized X12 schema specifically for two trading partners, X and Y) • Advanced shipment notices (uses the standard X12 schema)	You need four inbound backend integrations because you actually get the invoices in two different formats, one that is X12 standard and another that is custom. (You can collapse the four into a single backend integration, if that is your design preference.)

Document Types	How Many Backend Integrations Do You Need
You are expecting to send the following types of outbound documents to various trading partners: • Purchase orders (uses the standard X12 schema) • Purchase orders (uses a customized X12 schema specifically for two trading partners, X and Y) • Inventory inquiry (uses the standard X12 schema)	You need three outbound backend integrations because you are generating purchase orders in two different formats, one that is X12 standard and another that is custom. (You can collapse the three into a single backend integration, if that is your design preference.)

Inbound Message Processing

This section describes the high-level steps involved in processing an inbound B2B message.



Step	Description
1	A message sent or a file dropped in an FTP location by your trading partner arrives at the adapter endpoint (AS2 or FTP). It receives and unpacks the message.
2	The message is translated from EDI to a canonical XML format and persisted in the Oracle Integration persistence store. A unique ID is assigned to it.
3	Based on the current message type and the inbound agreements defined for the trading partner, an appropriate inbound backend integration is triggered. The message ID is handed to it.
4	The backend integration instance starts and receives the message ID at its REST trigger endpoint.
5	The backend integration, given the message ID, retrieves the translated canonical XML message from the Oracle Integration persistence store. It uses the B2B fetch message operation to retrieve it.
6	The canonical XML is further transformed to a backend application message.
7	Using an application adapter, the backend application message is sent to your backend application.

Step	Description
8	Your backend application now consumes the business transaction sent by your trading partner. Further processing is performed.

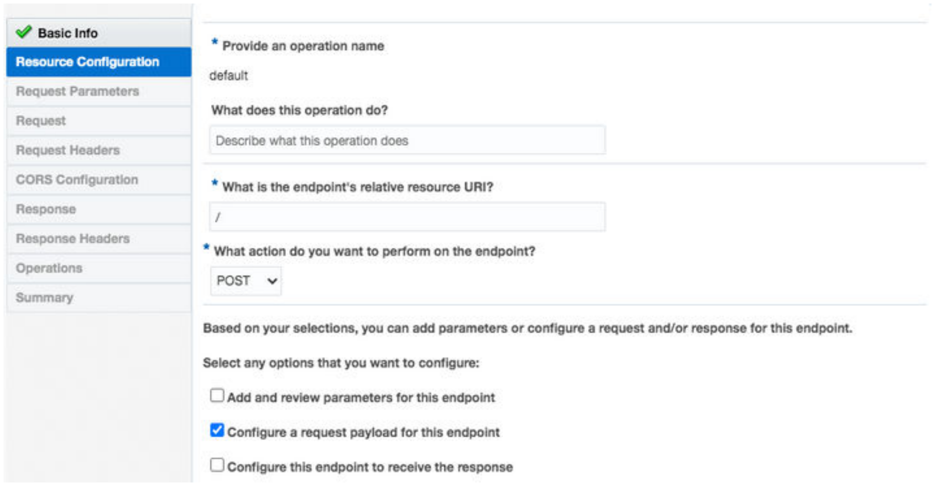
Design an Inbound Backend Integration

An inbound backend integration is triggered automatically by the B2B integration for receiving messages using a local integration invoke (that is, call an integration) action. For it to be triggered correctly, the backend integration must adhere to an API contract that requires:

- The integration must have a REST Adapter trigger configured for OAuth 2.0.
- The REST Adapter trigger must use `/` as the resource URI (that is, a root resource URI).
- The REST Adapter trigger must use a specific request payload schema, given in step 1(e).
- The REST Adapter trigger must not return a response; it must be asynchronous. This allows the B2B integration for receiving messages to be unblocked quickly and continue processing, even though the backend integration may take time to process.

This backend integration is given a collection of message IDs, in the form of repeating **b2b-message-reference** elements. A collection is necessary when your trading partner sends a batched message (that is, in one message or file, there may be multiple business documents). The B2B integration for receiving messages automatically splits the message into multiple documents and returns one **b2b-message-reference** element for each one. A maximum chunk of 200 records is provided at a time to the backend integration. If the inbound message has more documents, the backend integration may be called multiple times with chunks of 200.

1. Create the integration and configure the REST Adapter trigger.
   a. Select an App Driven Orchestration integration pattern. Give it any name. For this example, `Backend - Inbound Purchase Orders` is used).
   b. Add a REST Adapter trigger using the Sample REST Endpoint Interface (or using any other REST Adapter trigger connection that uses the OAuth 2.0 Or Basic Authentication security policy).
   c. Enter `Receive-B2B-Msg` (or any other name) as the name of the trigger connection.
   d. On the Resource Configuration page, enter `/` as the resource URI, select the **POST** action, and enable the **Configure a request payload for this endpoint** checkbox.



* Provide an operation name

default

What does this operation do?

Describe what this operation does

* What is the endpoint's relative resource URI?

/

* What action do you want to perform on the endpoint?

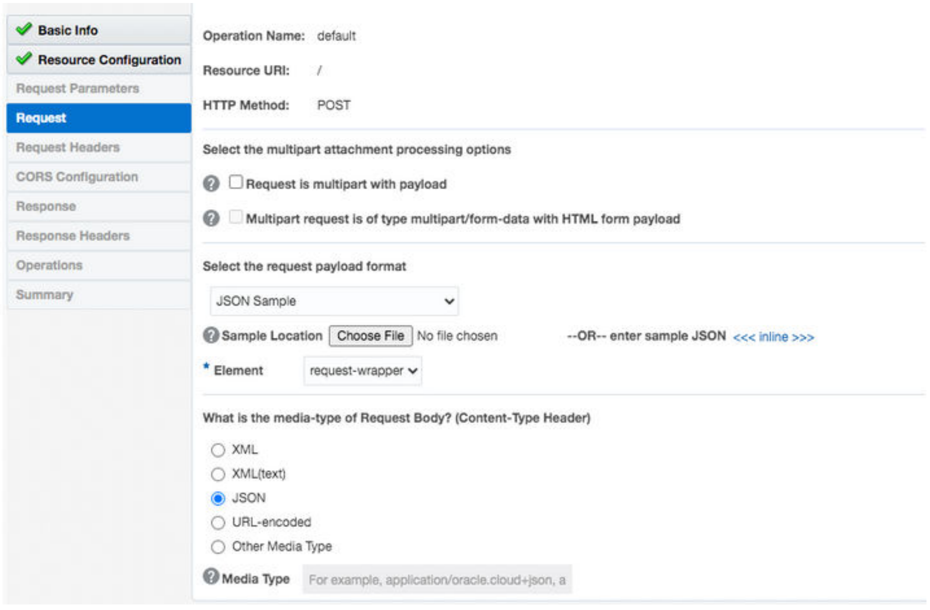
POST

Based on your selections, you can add parameters or configure a request and/or response for this endpoint.

Select any options that you want to configure:

- [ ] Add and review parameters for this endpoint
- [x] Configure a request payload for this endpoint
- [ ] Configure this endpoint to receive the response

e. On the Request Parameters page, select **JSON Sample** and paste the JSON provided below as an inline sample.



Operation Name: default

Resource URI: /

HTTP Method: POST

Select the multipart attachment processing options

- [ ] Request is multipart with payload
- [ ] Multipart request is of type multipart/form-data with HTML form payload

Select the request payload format

JSON Sample

Sample Location No file chosen --OR-- enter sample JSON <<< inline >>>

* Element request-wrapper

What is the media-type of Request Body? (Content-Type Header)

- XML
- XML(text)
- [✓ JSON]
- URL-encoded
- Other Media Type

Media Type For example, application/oracle.cloud+json, a

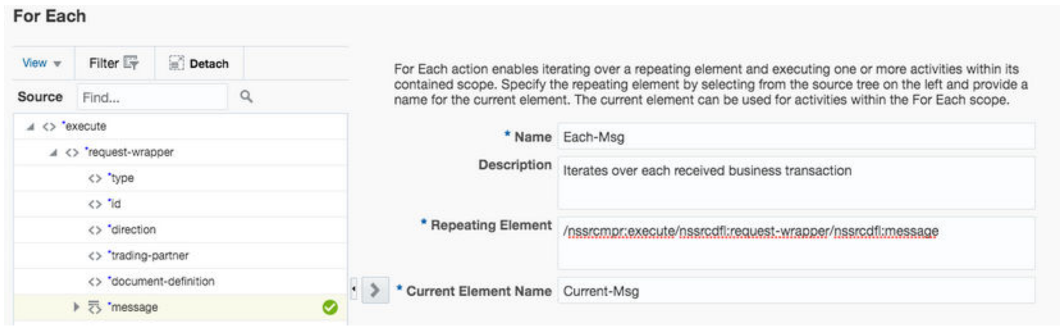
```
{
  "type": "MSG",
  "id": "12345",
  "direction": "INBOUND",
  "trading-partner": "ACME",
  "document-definition": "PO_850",
  "message": [
    {
      "b2b-message-reference":
"biz:0AC400D117503A8246000000347849EB"
    },
    {
      "b2b-message-reference":
"biz:0AC400D117503A8246000000347849EA"
```

```
            }
        ]
    }
```

> **Note:**
>
> In the above JSON sample, the structure is important. The values are just placeholders or representative values.

f. Click **Next** to access the Summary page to review your selections, then click **Done**.

2. Place a for-each action after the REST Adapter trigger.
3. Select **request-wrapper** > **message** as the **Repeating Element** and enter `Current-Msg` as the **Current Element Name** value.

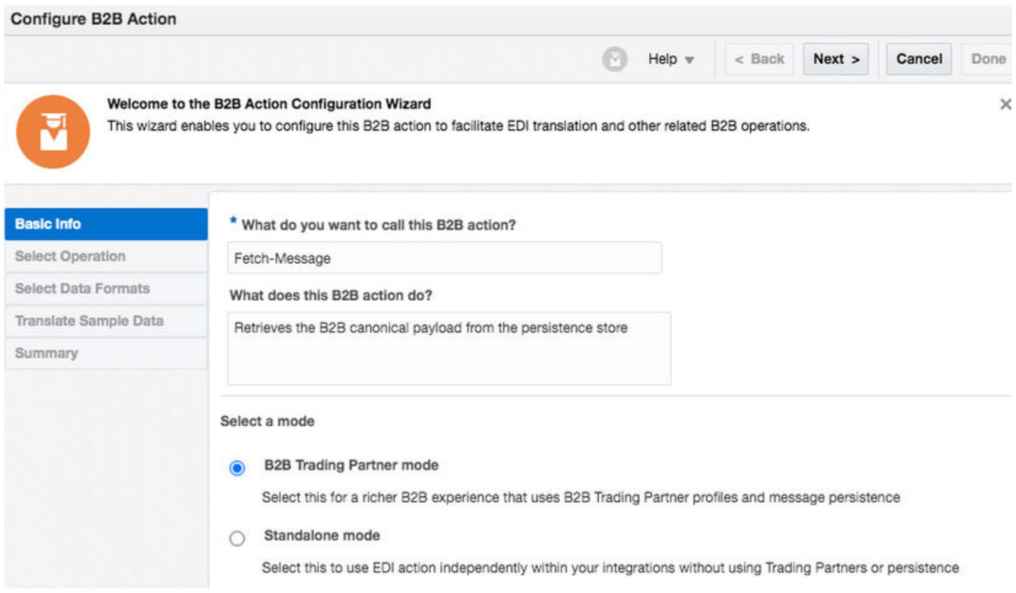


For Each action enables iterating over a repeating element and executing one or more activities within its contained scope. Specify the repeating element by selecting from the source tree on the left and provide a name for the current element. The current element can be used for activities within the For Each scope.

* Name	Each-Msg
Description	Iterates over each received business transaction
* Repeating Element	/nssrcmpr:execute/nssrcdfl:request-wrapper/nssrcdfl:message
* Current Element Name	Current-Msg

4. Add a scope inside the for-each action and name it (for this example, named `Handle-One-Message`).
5. Add a B2B action configured with a Fetch Message operation inside the scope. The B2B action appears in the palette under **Actions** > **Data**.

   a. Enter a name (for this example, named `Fetch-Message`), select **B2B Trading Partner mode**, and click **Next**.



Welcome to the B2B Action Configuration Wizard

This wizard enables you to configure this B2B action to facilitate EDI translation and other related B2B operations.

* What do you want to call this B2B action?

Fetch-Message

What does this B2B action do?

Retrieves the B2B canonical payload from the persistence store

Select a mode

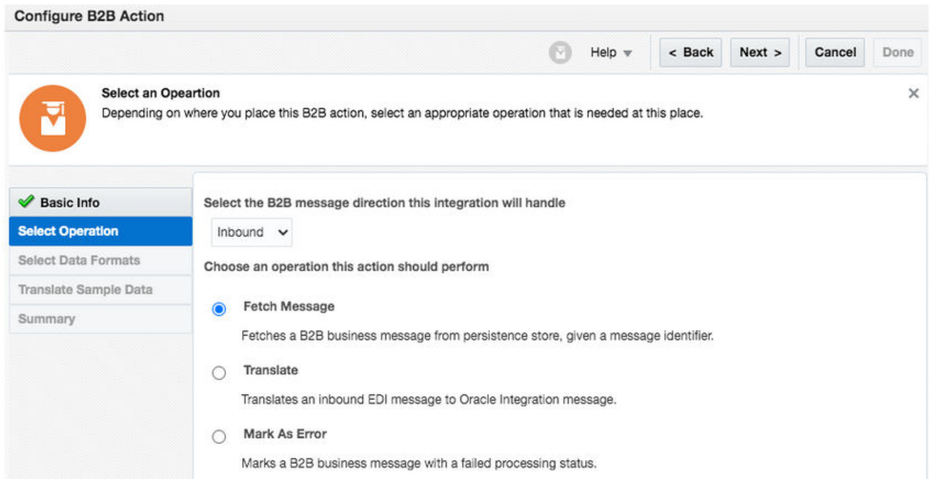
[✓ B2B Trading Partner mode]

Select this for a richer B2B experience that uses B2B Trading Partner profiles and message persistence

Standalone mode

Select this to use EDI action independently within your integrations without using Trading Partners or persistence

b. Select **Inbound** as the direction and **Fetch Message** as the operation. Click **Next**.



Configure B2B Action

Select an Opertion
Depending on where you place this B2B action, select an appropriate operation that is needed at this place.

Select the B2B message direction this integration will handle
Inbound

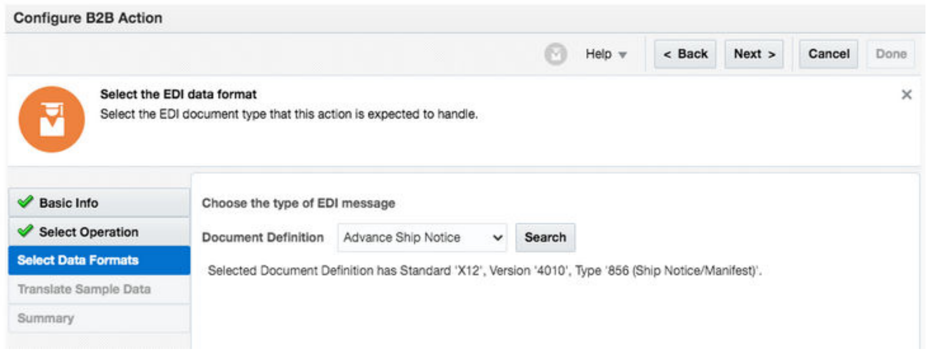
Choose an operation this action should perform

● Fetch Message
Fetches a B2B business message from persistence store, given a message identifier.

○ Translate
Translates an inbound EDI message to Oracle Integration message.

○ Mark As Error
Marks a B2B business message with a failed processing status.

c. On the Select Data Formats page, select a document definition from the drop-down list that this backend integration is handling.
Existing B2B documents are displayed in the drop-down list for selection. Alternately click **Search** to select a B2B document by document standard, version, and type.



Configure B2B Action

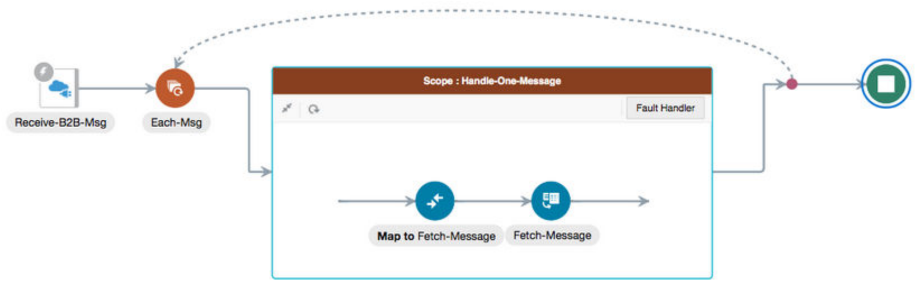
Select the EDI data format
Select the EDI document type that this action is expected to handle.

Choose the type of EDI message
Document Definition: Advance Ship Notice

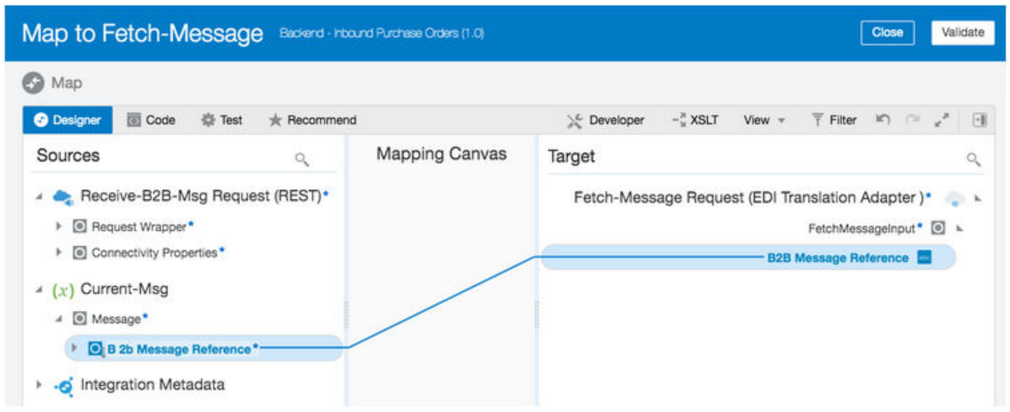
Selected Document Definition has Standard 'X12', Version '4010', Type '856 (Ship Notice/Manifest)'.

d. Click **Next** once you make the selection.

e. In the Summary Page, review your selections, and click **Done**.
The integration looks as follows.



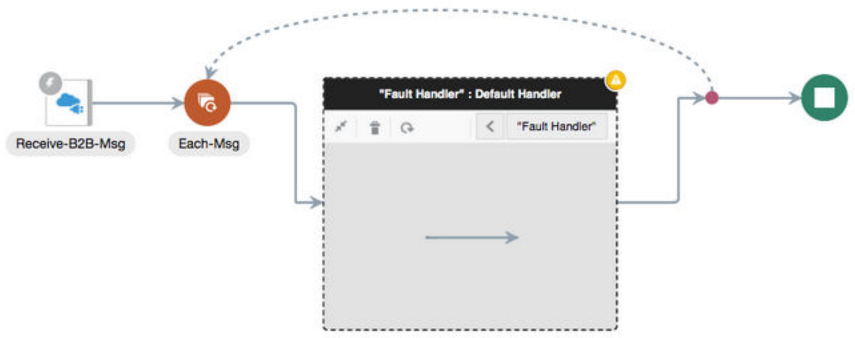
f. Configure the mapper to **Fetch-Message**.
   Expand **Current-Msg** and map its **B 2b Message Reference** to the **B2B Message Reference** of **FetchMessageInput**.



g. Close and apply changes to the mapper.

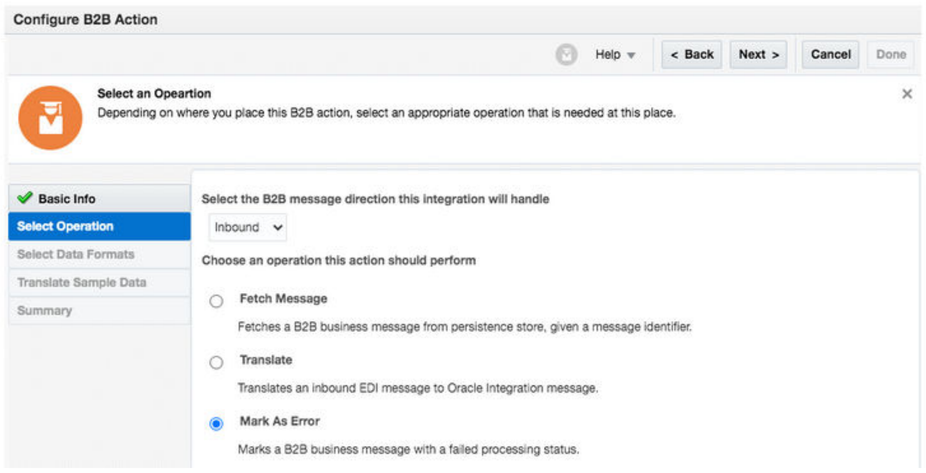
6. Add a scope level fault handler.
   This handler is added so that if the message fails to be processed by the backend application, the corresponding transaction is marked as **Failed** in the **B2B Tracking** > **Business Messages** view (instead of **Success**).

   a. Click **Fault Hander** > **Default Handler** for the scope. Initially it is empty, as shown below.



   b. Place a B2B action inside the fault handler and name it (for this example, `Mark-As-Error`). Click **Next**.

   c. Select **Inbound** as the direction and **Mark As Error** as the operation. Click **Next**.



Configure B2B Action

Help ▾ | < Back | Next > | Cancel | Done

Select an Opeartion
Depending on where you place this B2B action, select an appropriate operation that is needed at this place.

- ✓ Basic Info
- Select Operation
- Select Data Formats
- Translate Sample Data
- Summary

Select the B2B message direction this integration will handle
Inbound

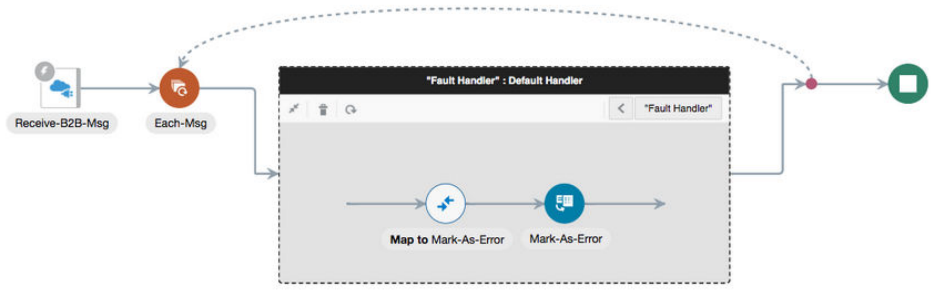
Choose an operation this action should perform

○ Fetch Message
Fetches a B2B business message from persistence store, given a message identifier.

○ Translate
Translates an inbound EDI message to Oracle Integration message.

● Mark As Error
Marks a B2B business message with a failed processing status.

d. On the Summary page, review your selections, and click **Done**.
The fault handler now looks as follows.

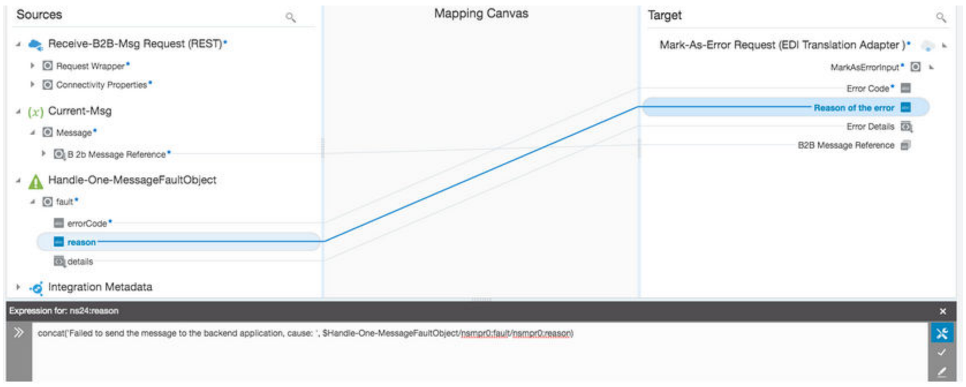


e. Configure the mapper to **Mark-As-Error** by mapping the following elements:

- Source **Current-Msg > Message** > **B 2b Message Reference** to target **MarkAsErrorInput** > **B2B Message Reference**.
- Source **Handle-One-MessageFaultObject** > **errorCode** to target **MarkAsErrorInput** > **Error Code**.
- Source **Handle-One-MessageFaultObject** > **reason** to target **MarkAsErrorInput** > **Reason of the error**, and in the Expression Builder, enter:

```
concat('Failed to send the message to the backend
application, cause: ', $Handle-One-MessageFaultObject/
nsmpr0:fault/nsmpr0:reason)
```

- Source **Handle-One-MessageFaultObject** > details to target **MarkAsErrorInput** > **Error Details**.



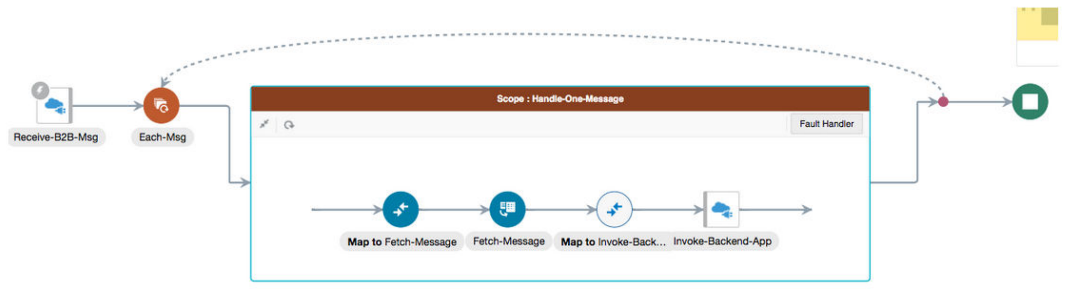
Expression for: ns24:reason

```
concat('Failed to send the message to the backend application, cause: ', $Handle-One-MessageFaultObject/nsmpr0:fault/nsmpr0:reason)
```

f. Save and apply the mapping.

g. Exit the fault handler by clicking .

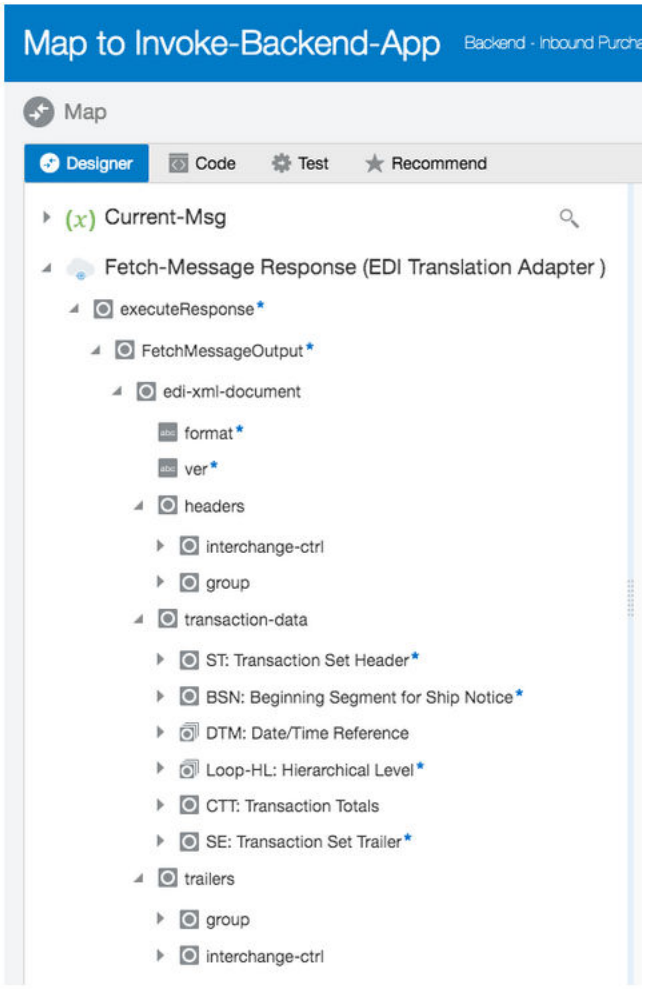
7. Add actions to call a backend application.
   Add one or more actions inside the scope to send the business message to a backend application.



In the example above, a REST Adapter invoke connection sends the message to a backend application. To use with your specific backend application, you have many application adapters provided in Oracle Integration that can interface with several popular backend applications. You can also use technology adapters, including REST, SOAP, JMS, AQ, File, and so on.

The **Fetch-Message Response** (EDI Translation Adapter) provides you with a B2B canonical XML from which to map. Its **edi-xml-document** is the key element that contains the canonical form of the inbound EDI document. See Schema Elements for Inbound EDI.

You must design the data mapping to prepare the message prior to the backend invoke. This can be a complex task. On the left-hand side is the B2B canonical XML format, represented by the **edi-xml-document**. On the right-hand side (not shown below) is your backend application schema for the business document. You must create mappings for the elements on the left-hand side (B2B canonical XML) to the right-hand side (backend application schema). The mappings cannot be generalized since the right-hand side is specific to a target backend application.



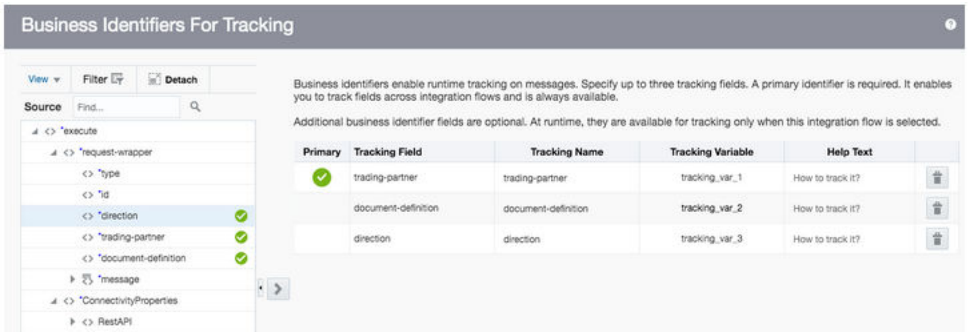
Map to Invoke-Backend-App Backend - Inbound Purcha

Map

Designer | Code | Test | Recommend

- (x) Current-Msg
- Fetch-Message Response (EDI Translation Adapter)
  - executeResponse*
    - FetchMessageOutput*
      - edi-xml-document
        - format*
        - ver*
        - headers
          - interchange-ctrl
          - group
        - transaction-data
          - ST: Transaction Set Header*
          - BSN: Beginning Segment for Ship Notice*
          - DTM: Date/Time Reference
          - Loop-HL: Hierarchical Level*
          - CTT: Transaction Totals
          - SE: Transaction Set Trailer*
        - trailers
          - group
          - interchange-ctrl

8. Add identifiers for integration tracking.
   Select fields from the input schema for integration tracking to complete the integration.



Business Identifiers For Tracking

View | Filter | Detach

Source Find...

- <> *execute
  - <> *request-wrapper
    - <> *type
    - <> *id
    - <> *direction ✓
    - <> *trading-partner ✓
    - <> *document-definition ✓
    - *message
  - <> *ConnectivityProperties
    - <> RestAPI

Business identifiers enable runtime tracking on messages. Specify up to three tracking fields. A primary identifier is required. It enables you to track fields across integration flows and is always available.

Additional business identifier fields are optional. At runtime, they are available for tracking only when this integration flow is selected.

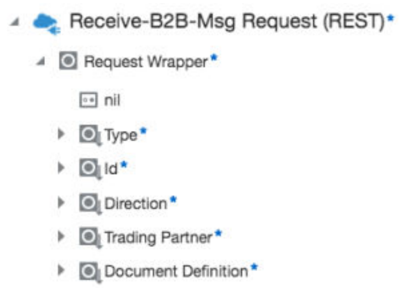
Primary	Tracking Field	Tracking Name	Tracking Variable	Help Text
✓	trading-partner	trading-partner	tracking_var_1	How to track it?
	document-definition	document-definition	tracking_var_2	How to track it?
	direction	direction	tracking_var_3	How to track it?

9. Save and then activate the backend integration.

Design a Backend Integration to Handle Multiple Types of Documents

The detailed steps assume you create one backend integration for each document definition that you want to handle. You can clone the integration for other document types because the basic pattern is identical.

If you want to design a single integration that handles multiple document types, add a switch action in your integration and specific routes for each document definition or trading partner. The request schema provides these elements.



```
Receive-B2B-Msg Request (REST)*
  Request Wrapper*
    nil
    Type*
    Id*
    Direction*
    Trading Partner*
    Document Definition*
```

Each route has its own B2B Fetch Message operation, a data mapping, and an invocation call to the backend application. This design pattern is a trade-off between ease of development and operations. For example, if you need to fix a mapping for your purchase order document, you must:

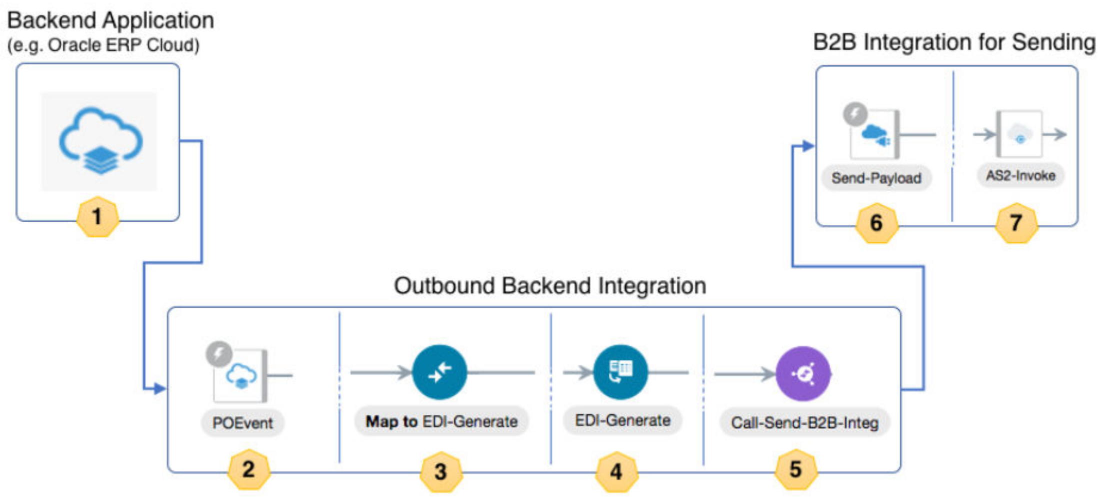
- Deactivate the backend integration that handles the purchase order document.
- Fix the mapping.
- Reactivate the integration.

If you had separate backend integrations for each document type, you are isolating the impact to one specific integration. Whereas, if everything is built into a single integration, you have a wider impact in a production deployment, for example.

Outbound Message Processing

This section describes the high-level steps involved in processing an outbound B2B message.

The high-level steps involved in processing an outbound B2B message are as follows.



Step	Description
1	Your backend application has a requirement to send a business transaction to an external trading partner. It triggers your outbound backend integration by sending it a notification message.
2	Your backend integration instance receives the notification that includes the application message in the backend application message format.
3	Using a mapper, the backend application message is transformed into a B2B canonical XML format.
4	The canonical XML message is provided to a B2B action for outbound translation (the action named **EDI-Generate** above). A trading partner is specified as an input to the B2B action. The B2B action translates the canonical XML message to a native EDI format (X12 or EDIFACT) and persists it in the Oracle Integration persistence store. A unique ID is assigned to it.
5	Based on the target trading partner, the current document type, and the outbound agreements defined for the trading partner, an appropriate B2B integration for sending messages is triggered. The message ID is handed to it.
6	The B2B integration for sending messages instance starts and receives the message ID at its REST Adapter trigger endpoint.
7	The B2B integration for sending messages instance uses an adapter (AS2 or FTP) to pack the message and then transmit it to the external trading partner through the AS2 or FTP protocol.

Design an Outbound Backend Integration

An outbound backend integration is triggered by a backend application. For sending a message to an external trading partner, this integration must know exactly which trading partner the message must go to. This is specified either directly or indirectly by the backend application in either of two ways:

1. The trading partner's identifier is specified. It is the same identifier you entered when you created a new trading partner in B2B, displayed in the Primary Information page.
   or
2. An application partner ID is specified. This is a type of B2B identifier you can add to a trading partner.

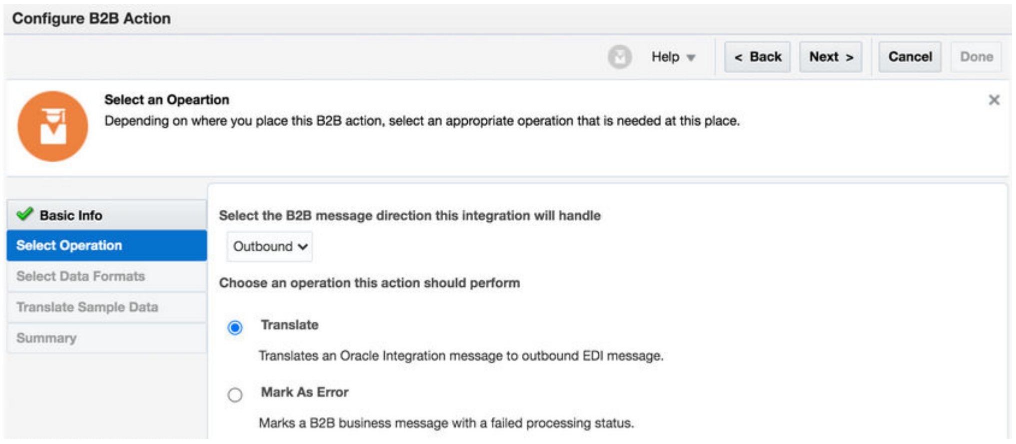
The idea behind the Application Partner ID is the following: the backend application and the B2B system both model the concept of a trading partner. A backend application may treat such a business entity in a specific role, such as a supplier, vendor, purchaser, and so on. The business entity is assigned a unique identification in the backend application. However, in B2B, you again register this entity as a B2B trading partner for message exchange purposes. Your suppliers, vendors, purchases are all treated the same: as B2B trading partners. At the hand-off point between the backend application and B2B, one of these systems must map between the backend application's entity ID and the B2B system's trading partner ID. The application partner ID is exactly for this purpose; it allows you to add the backend application's entity ID into the B2B system.

For this to work, add the Application Partner ID as a B2B identifier type to all of your trading partners. The value is the unique identification defined in your backend application. Once you define these, the B2B system knows how to look up a trading partner, given the value for Application Partner ID.

Step 3 specifies how you map the application partner ID element in the TranslateInput schema.

1. Create the integration and configure the trigger connection.
   Depending on how you want a notification from the backend application to arrive, configure a trigger connection using an appropriate application adapter or a technology adapter.

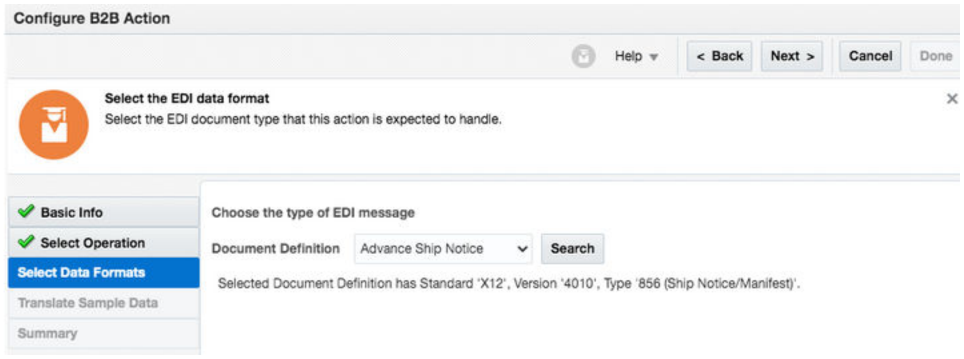
   In our example, a simple REST Adapter trigger is used. Replace it with Oracle ERP Cloud Adapter, Oracle NetSuite Adapter, or another one.
2. Place a B2B action with an outbound translate operation.
   a. Add a B2B action into the integration. The B2B action can be seen in the palette under **Actions** > **Data**.
   b. Enter a name (for this example, `EDI-Generate` is added). Click **Next**.
   c. Select **Outbound** as the direction and **Translate** as the operation. Click **Next**.



Select an Opeartion
Depending on where you place this B2B action, select an appropriate operation that is needed at this place.

Select the B2B message direction this integration will handle
Outbound
Choose an operation this action should perform
(●) Translate
Translates an Oracle Integration message to outbound EDI message.
( ) Mark As Error
Marks a B2B business message with a failed processing status.

   d. In the Select Data Formats page, select a document definition from the drop-down list for this backend Integration to handle. Click **Next** after you make the selection. Existing B2B documents are displayed in the drop-down list for selection. Alternately, click **Search** to select a B2B document by document standard, version, and type.

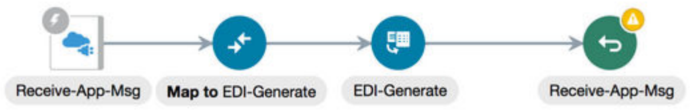


Select the EDI data format
Select the EDI document type that this action is expected to handle.

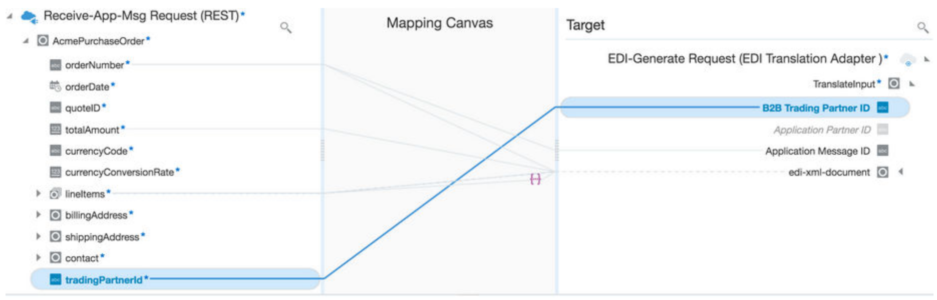
Choose the type of EDI message
Document Definition: Advance Ship Notice

Selected Document Definition has Standard 'X12', Version '4010', Type '856 (Ship Notice/Manifest)'.

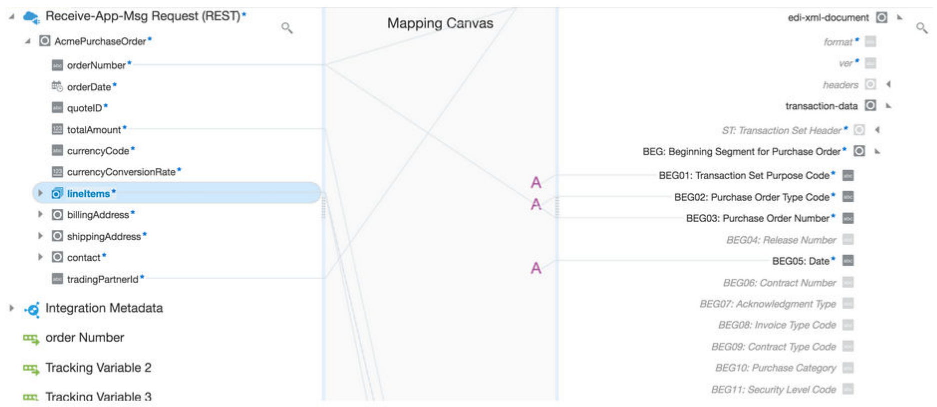
   e. In the Summary Page, review your selections and then click **Done**.
      Your integration looks as follows.



3. Configure the mapper to convert the application message to B2B canonical format. This mapping can be complex. Let us break it down into two parts.
   - In the first part, map an element to either **B2B Trading Partner ID** or **Application Partner ID**. These elements are used to specify the trading partner to which to send. This concept was explained earlier. In the example shown, only the **B2B Trading Partner ID** has been mapped.
     Also, optionally map **Application Message ID**. This is a message ID assigned by the backend application, if any. If this is mapped, the value is displayed in **Monitoring** > **B2B Tracking** > **Business Messages** in the details panel for any outbound message. Its value is not used during processing, but is stored and is displayed for your reference only.

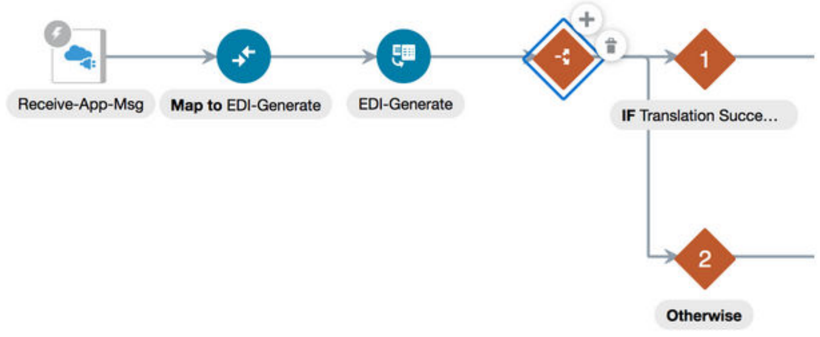


   - The second part of the map is for **edi-xml-document**, which is the B2B canonical XML format for an EDI document, shown on the right-hand side of the map. On the left-hand side is your backend application schema for the business document. Because that schema is very specific to your backend application, the mappings from the backend application schema to the B2B canonical XML cannot be generalized.
     The following example maps a custom structure, an **AcmePurchaseOrder** to the **edi-xml-document**, which represents the canonical format for an EDI X12, 850, Purchase Order.



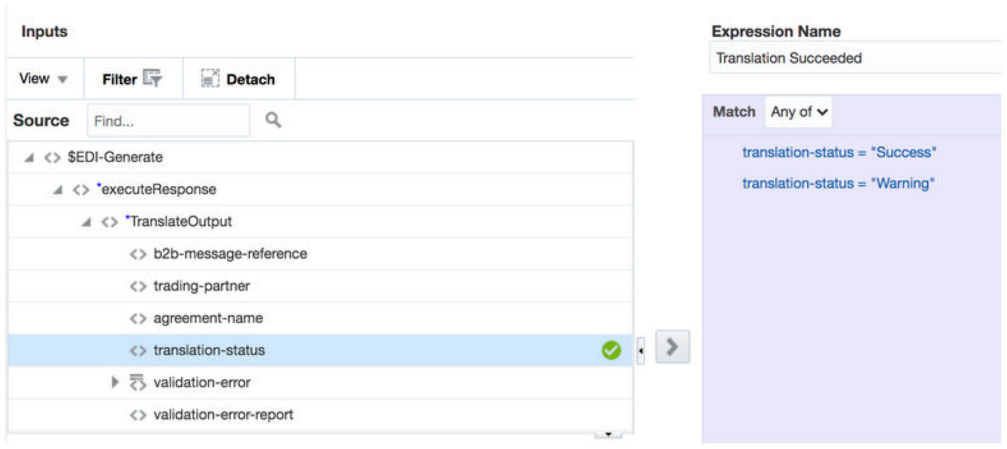
4. Add a check for successful EDI translation.

a. Place a switch action after the B2B translate (that is, **EDI-Generate**) action. This action is for checking translation status to find if the EDI translation succeeded or failed.



b. Add a route, called **Translation Succeeded**, and configure the following expression:

```
translation-status = "Success" or translation-status = "Warning"
```



Expression Name
Translation Succeeded

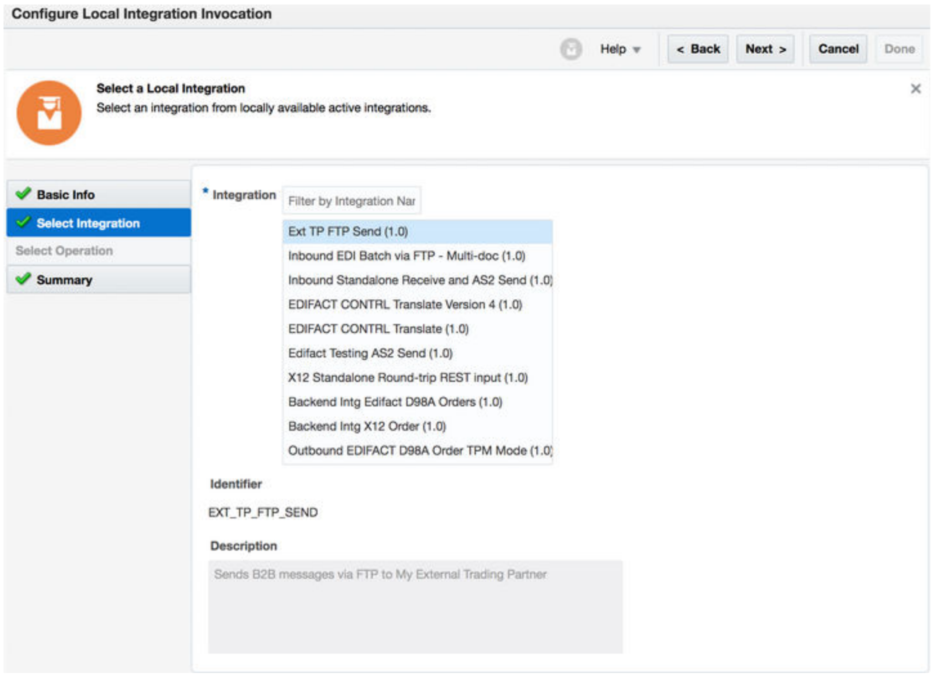
Match Any of

translation-status = "Success"
translation-status = "Warning"

5. If translation is successful, prepare to deliver the document to the trading partner.

   a. Add a local integration invoke (that is, call an integration) action in the **Translation Succeeded** route.

   b. On the Select Integration page, select any available B2B Integration for sending messages (the exact name of such an integration varies, but the name always ends with **AS2 Send** or **FTP Send**). In the following example, the selected integration, **Ext TP FTP Send (1.0)** is a B2B integration for sending messages. While the selection must be a B2B integration for sending messages because of a specific API contract, it does not matter which specific trading partner it is, since this selection is overridden from the mapper as explained in Step c .


Configure Local Integration Invocation

Help ▾ | < Back | Next > | Cancel | Done

Select a Local Integration
Select an integration from locally available active integrations.

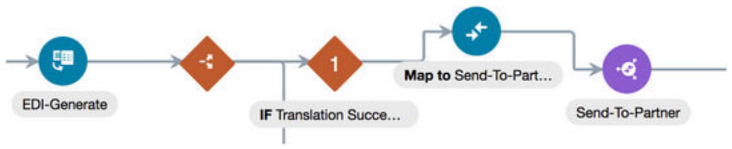
* Integration: Filter by Integration Nar

- Ext TP FTP Send (1.0)
- Inbound EDI Batch via FTP - Multi-doc (1.0)
- Inbound Standalone Receive and AS2 Send (1.0)
- EDIFACT CONTRL Translate Version 4 (1.0)
- EDIFACT CONTRL Translate (1.0)
- Edifact Testing AS2 Send (1.0)
- X12 Standalone Round-trip REST input (1.0)
- Backend Intg Edifact D98A Orders (1.0)
- Backend Intg X12 Order (1.0)
- Outbound EDIFACT D98A Order TPM Mode (1.0)

Identifier
EXT_TP_FTP_SEND

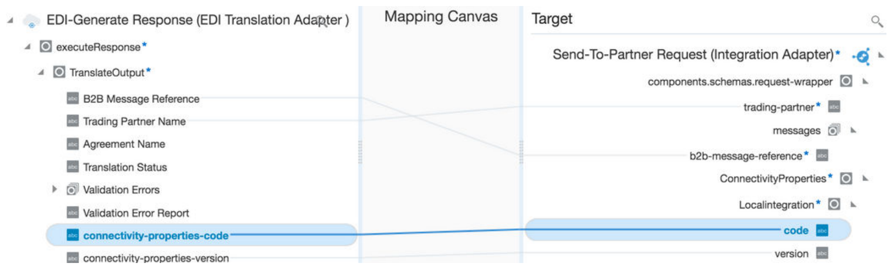
Description
Sends B2B messages via FTP to My External Trading Partner

c. Click **Next** several times, and then click **Done**. The **Translation Succeeded** route now looks as follows:



d. Configure the mapper before the local integration invoke (that is, map to **Send-To-Partner**) as follows:

- Map source **TranslateOutput** > **B2B Message Reference** to target **components.schemas.request-wrapper** > **messages** > **b2b-message-reference**.
- Map source **TranslateOutput** > **Trading Partner Name** to target **components.schemas.request-wrapper** > **trading-partner**.
- Map source **TranslateOutput** > **connectivity-properties-code** to target **ConnectivityProperties** > **LocalIntegration** > **code**.
- Map source **TranslateOutput** > connectivity-properties-version to target **ConnectivityProperties** > **LocalIntegration** > **version**.



The mappings for **ConnectivityProperties** > **LocalIntegration** > **code** and **version** override the selection of an integration in Step 5(b)).

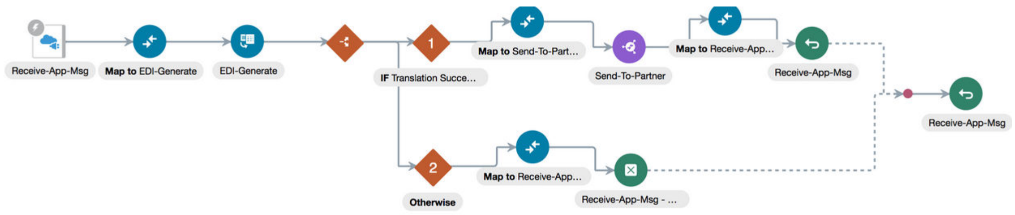
Here's how the routing works:

- The B2B action checks the trading partner's outbound agreements and finds one that matches the B2B document selected in step 2(d).
- The transport linked to the outbound agreement is found.
- Once the transport is known, the B2B integration for sending messages linked to the transport is also known. That's the integration to call for delivering the document to the trading partner. The B2B action populates the **connectivity-properties-code** and **connectivity-properties-version** with that integration's identifier and version.
- If you have mapped the fields correctly, the local integration invoke (that is, calls an integration) action honors the mapper override and calls the integration specified in the **connectivity-properties-code** and **connectivity-properties-version**, instead of the one selected in Step 5(b).

6. Add a return action for success and a fault for an error.

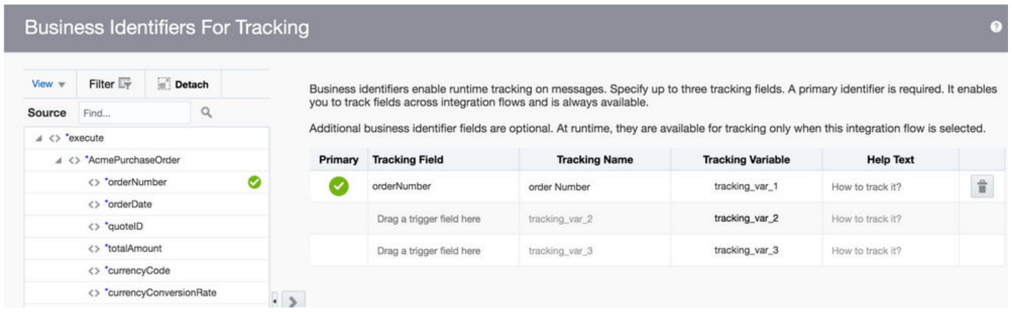
   a. Add a return action for the integration to return a successful response for the **Translation Succeeded** route and a fault return for the **Otherwise** route.
   This step is specific to the example selected with the REST Adapter trigger. Your case may vary. Depending on your backend application, there may or not be a mechanism to relay the outcome of a successful or a failed translation back into the backend application.

   The completed outbound integration looks as follows.



7. Add B2B business identifiers for integration tracking.

   a. Select fields from the input schema for integration tracking to complete the integration. This is largely specific to your backend application schema. The tracking configuration for this example integration is as follows, but your integration can look different.



Business identifiers enable runtime tracking on messages. Specify up to three tracking fields. A primary identifier is required. It enables you to track fields across integration flows and is always available.

Additional business identifier fields are optional. At runtime, they are available for tracking only when this integration flow is selected.

Primary	Tracking Field	Tracking Name	Tracking Variable	Help Text
✓	orderNumber	order Number	tracking_var_1	How to track it?
	Drag a trigger field here	tracking_var_2	tracking_var_2	How to track it?
	Drag a trigger field here	tracking_var_3	tracking_var_3	How to track it?

b. Save, and then activate the backend integration.

Design a Backend Integration to Handle Multiple Types of Documents

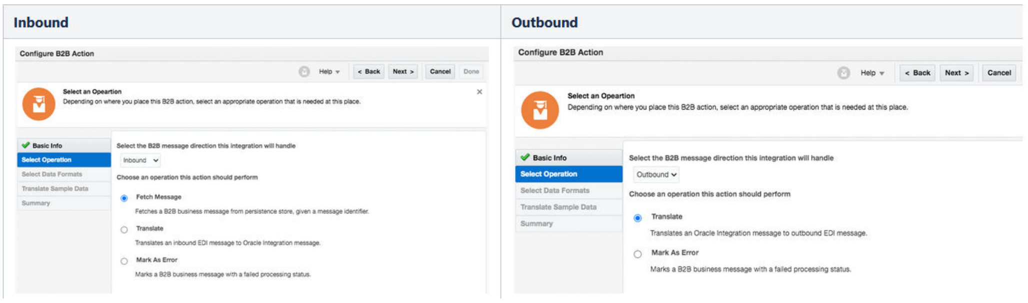
The detailed steps assume you create one backend integration for each document definition you want to handle. You may clone the integration for other document types because the basic pattern is identical.

Similar to an inbound backend integration, you may design a single integration that handles multiple document types by adding a switch action in your integration and specific routes for each document definition or trading partner and their own map and B2B translate action.

Use the B2B Action In Trading Partner Mode

This section describes the operations provided by the B2B action during trading partner mode configuration.

The following operations are available in the inbound and outbound directions.



Inbound

Configure B2B Action

Select an Opeartion
Depending on where you place this B2B action, select an appropriate operation that is needed at this place.

Select the B2B message direction this integration will handle: Inbound

Choose an operation this action should perform

[✓ Fetch Message]
Fetches a B2B business message from persistence store, given a message identifier.

Translate
Translates an inbound EDI message to Oracle Integration message.

Mark As Error
Marks a B2B business message with a failed processing status.

Outbound

Configure B2B Action

Select an Opeartion
Depending on where you place this B2B action, select an appropriate operation that is needed at this place.

Select the B2B message direction this integration will handle: Outbound

Choose an operation this action should perform

[✓ Translate]
Translates an Oracle Integration message to outbound EDI message.

Mark As Error
Marks a B2B business message with a failed processing status.

Of these five operations, you must use the three highlighted below in your backend integrations. Those are the only ones you need to more fully understand. The remaining operations are used in the B2B transport integrations (that is, B2B integrations for receiving messages and sending messages). Because those integrations are automatically created, you don't need to understand their usage in as much detail.

Inbound Direction

Operation	Used By	Purpose
Fetch Message	Inbound backend integration	This operation retrieves an already processed B2B business message from the Oracle Integration persistence store. It outputs the B2B canonical XML format for a business message, given **b2b-message-reference** as input. The canonical XML format is represented by the **edi-xml-document** element. It is accessible inside an inbound backend integration. You use the mapper to transform it into a backend application format. You must select a specific B2B document during design time when you configure this operation. At runtime, it only retrieves a compatible document. If this operation is given a **b2b-message-reference** for a different B2B document, an error occurs (for example, if the fetch message is configured for a purchase order and at runtime it was asked to retrieve an invoice).
Translate	B2B integration for receiving messages	The B2B integration for receiving messages uses this operation for parsing and debatching an inbound EDI message into B2B canonical XML format, represented by the **edi-xml-document** element. One inbound EDI message may produce multiple B2B business messages (each one having a separate canonical XML document). The action outputs a collection of repeating **b2b-message-reference** elements, each containing an internal message ID of one business message. The canonical XML format is accessible inside the integration with the fetch message operation.
Mark As Error	Inbound backend integration	This operation provides for more robust error handling, in case of failures. This operation updates a B2B business message and reflects the failure to process this message by the backend integration, if an error occurs.

Outbound Direction

Operation	Used By	Purpose
Translate	Outbound backend integration	You must use this action directly within your outbound backend integrations. An outbound backend integration uses this operation to translate from a B2B canonical XML format to an EDI format. The EDI format cannot be accessed inside the integration directly. Instead, an internal message ID is assigned that is returned in the element **b2b-message-reference**. You can view or download the EDI-formatted payload from **Monitoring** > **B2B Tracking** > **Business Message** or by using the B2B Monitoring REST APIs.
Mark As Error	B2B integrations for receiving and sending messages	This operation provides for more robust error handling, in case of failures. This operation updates a B2B wire message and reflects the failure to process this message by the B2B integration for sending messages. For example, for the FTP sending messages integration, if the file write operation fails, this operation updates the wire message as failed. There is also a similar error condition that can occur while sending back a functional acknowledgment in the B2B integration for receiving.

3

Use B2B for Oracle Integration in Standalone Mode

Topics:

Standalone Mode Concepts

B2B for Oracle Integration provides several components that enable you to design integrations to handle B2B messages. This includes the AS2 Adapter and the B2B action for EDI translation. Use these components to create integrations to receive and send messages from (or to) your external trading partners. In this mode, the external trading partner is not defined explicitly as you do in B2B trading partner mode. Standalone mode ignores any trading partner definitions.

Standalone mode supports the following:

- Interfacing with external trading partners using any adapter provided in Oracle Integration: AS2, FTP, REST, SOAP, AQ and more.
- Handling any data formats that Oracle Integration supports: EDI, CSV, XML, JSON, and more.

In contrast, B2B trading partner mode currently supports a limited set of adapters (AS2 and FTP) and only EDI data formats.

B2B Action

The B2B action was previously called EDI Translate. This action was renamed because it now provides additional operations besides EDI translation. However, those operations are only available in B2B trading partner mode. When using the B2B action in standalone mode, it provides the two key operations, inbound and outbound EDI translation, just as it did before. Any existing integration you created using the EDI Translate action works the same as before.

Support for the UN/EDIFACT document standard is now available, in addition to X12, in both standalone mode and B2B trading partner mode.

Details about how to use the B2B action in your integrations are provided. See Use the B2B Action in Standalone Mode.

Adapters for Communication

You can design your B2B integrations with any adapter as a trigger or invoke connection for receiving or sending messages from (or to) a trading partner. This section provides pointers for using the popular communication protocols in conjunction with B2B processing below:

- AS2
  You can design an integration to receive or send EDI or another type of payload over the AS2 protocol. See Implement Common Patterns Using the AS2 Adapter.
- FTP
  You can design an integration to poll for files or send files with EDI or another type of payload. A stage file action is typically used with the FTP Adapter. See Process an Inbound EDI Batch File Using the Stage File Action.
- REST, plain HTTP
  You can design an integration to receive an EDI or another type of payload over REST or plain HTTP using the REST Adapter. For EDI processing, you can configure the REST Adapter trigger to accept a binary payload and process it using the B2B action. See Create a Simple Integration Using B2B for Oracle Integration.
- SOAP
  You can design an integration to receive or send a SOAP message from (or to) a trading partner, typically with an XML payload to cover any of the XML-based B2B standards. In this pattern, you bring the XSD schemas and upload them into the SOAP Adapter trigger or invoke adapter configuration.

Use the B2B Action in Standalone Mode

You can translate a message to or from the Electronic Data Interchange (EDI) format in an orchestrated integration with the B2B action.

The B2B action translates an incoming EDI document into an Oracle Integration XML message and an outgoing Oracle Integration XML message into an EDI X12 document.

Note:

This section provides an overview of how to add a B2B action to an orchestrated or scheduled integration. Complete integration design details are provided. See Create Application-Driven Orchestrated Integrations in *Using Integrations in Oracle Integration Generation 2*.

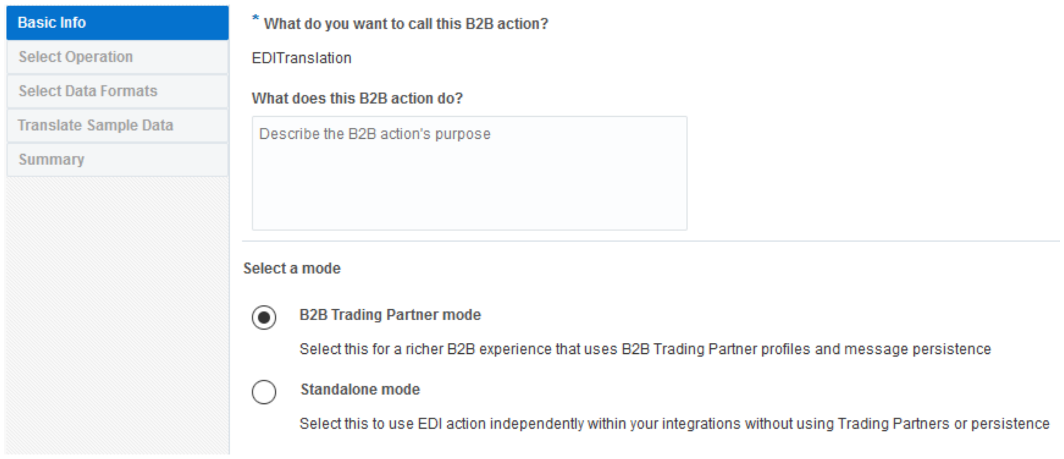
1. From the left navigation pane of the Oracle Integration home page, select **Integrations**.
2. Click **Create**, then select to create an **App Driven Integration** or **Scheduled Integration**.
3. Design your integration. For example, add a trigger connection if you created an **App Driven Integration**.
4. Add a B2B action to the integration in either of the following ways:
   - On the right side of the canvas, click and drag the **B2B** icon to the appropriate location.
   - Click at the location where you want to add the action, then select **B2B**.

   The Configure B2B Action Wizard is displayed.

5. Enter a name and optional description.
6. Select how you want to use the B2B action, then click **Next**.
   - **B2B Trading Partner Mode**: Select to include B2B trading partner profiles and message persistence in your integration.

     Note:

     This option is only available with Oracle Integration Generation 2.
   - **Standalone Mode**: Select to use the B2B action independently in your integration *without* a B2B trading partner profile or message persistence.



* What do you want to call this B2B action?

EDITranslation

What does this B2B action do?

Describe the B2B action's purpose

Select a mode

(●) B2B Trading Partner mode

Select this for a richer B2B experience that uses B2B Trading Partner profiles and message persistence

( ) Standalone mode

Select this to use EDI action independently within your integrations without using Trading Partners or persistence

7. If you select **B2B Trading Partner Mode**, follow these wizard steps:
   a. Select the B2B message direction for this B2B action:
      - **Inbound**: The B2B message is sent from the trading partner to the host trading partner (partner where Oracle Integration is installed).
      - **Outbound**: The B2B message is sent from the host trading partner (partner where Oracle Integration is installed) to the trading partner.

   b. Select the operation for the B2B action to perform based on the direction.

If You Selected...	Then Select an Operation...
Inbound	• **Fetch Message**: Fetches a B2B business message from the persistence store, given a message identifier. • **Translate**: Translates an inbound EDI message to an Oracle Integration message. • **Mark As Error**: Marks a B2B business message with a failed processing status.
Outbound	• **Translate**: Translates an Oracle Integration message to an outbound EDI message. • **Mark As Error**: Marks a B2B business message with a failed processing status.

c. If you select either of the following, the Select Data Format page is displayed.
   - **Inbound** message direction and **Fetch Message** operation
   - **Outbound** message direction and **Translate** operation

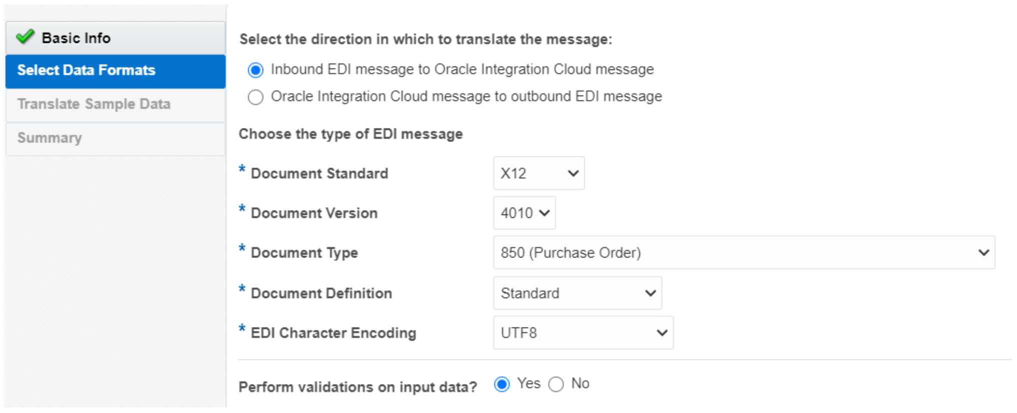
   i. Select the document definition for the B2B action to handle, and click **Search** to refresh the page.
   ii. Select the document standard, version, and type.
   iii. Click **Next**.

d. Review your selections on the Summary page, then click **Done**.

8. If you select **Standalone Mode**, follow these wizard steps:

   a. Specify the message translation and document format details.

Element	Description
Select the direction in which to translate the message	• **Inbound EDI message to Oracle Integration message**: When an integration receives an EDI document from a business partner, it is considered an inbound document (an EDI document is translated to XML). • **Oracle Integration message to outbound EDI message**: When an integration sends an EDI document to a business partner, it is considered an outbound document (an EDI document is generated from XML).
Document Standard	The document standard identifies the business protocol to follow when exchanging business documents between partners. • EDIFACT • X12
Document Version	Lists the supported versions of the selected document standard. Select the version to use.
Document Type	Select the document type (for example, purchase order, invoice, shipping notice, or others). The document types available for selection are based on the document version you selected.
Document Definition	Select the document definition (either **Standard** or a custom document definition that you created on the B2B Documents page). See Create a Custom B2B Document Definition.
EDI Character Encoding	Select the character encoding that the inbound EDI document is expected to use.
Perform validations on input data	• **Yes**: Validates the structure and data of an inbound EDI message. Enabling message validation has an impact on performance. If errors are found, translation does not succeed. • **No**: Errors are ignored during translation and the message is passed through in its current format.



Select the direction in which to translate the message:

[✓ Inbound EDI message to Oracle Integration Cloud message]
Oracle Integration Cloud message to outbound EDI message

Choose the type of EDI message

* Document Standard	X12
* Document Version	4010
* Document Type	850 (Purchase Order)
* Document Definition	Standard
* EDI Character Encoding	UTF8

Perform validations on input data? [✓ Yes] No

b. Click **Next**.

c. Specify to optionally upload sample data to test that translation is successful.

Element	Description
Select sample data to upload	Click **Browse** to upload a sample. To test inbound EDI message translation, upload an EDI document. To test outbound EDI message generation, upload an XML document.
Translate	Click to translate your sample data. Output is displayed in the **Output of translation** field. Any errors are displayed in the **Error in translation** field.

d. Click **Next**.

e. Review your selections on the Summary page, then click **Done**.

Develop Flows in B2B for Oracle Integration

Learn how to develop different types of B2B integration flows using the B2B action.

Topics:

- Parse an incoming EDI X12 document. Create a Simple Integration Using B2B for Oracle Integration.
- Process an inbound EDI batch file and send a functional acknowledgment. Process an Inbound EDI Batch File Using the Stage File Action.
- Transform an inbound EDI message to the XML format. Parse and Transform an Inbound EDI Message
- Create an EDI X12 document from an XML message. Generate Outbound EDI from an Application Message

Create a Simple Integration Using B2B for Oracle Integration

Using a simple example, let's explore how to use the B2B action in your integration to parse an incoming EDI X12 document.

Topics:

Create an Integration

Let's create a basic, inbound integration flow that receives an EDI document through a REST request, parses and validates the EDI, converts it to XML, and returns the XML in the response.

> **Note:**
>
> This integration flow uses REST for simplicity. You can substitute the REST Adapter trigger connection with any other adapter, such as the FTP Adapter, available in Oracle Integration. Generally, FTP is the transport type used with EDI.

1. In the navigation pane, click **Integrations**.
2. On the Integrations page, click **Create**.
3. Select **App Driven Orchestration** as the style to use. The Create New Integration dialog is displayed.
4. In the **What do you want to call your integration?** field, enter `Inbound EDI via REST`, then click **Create**.

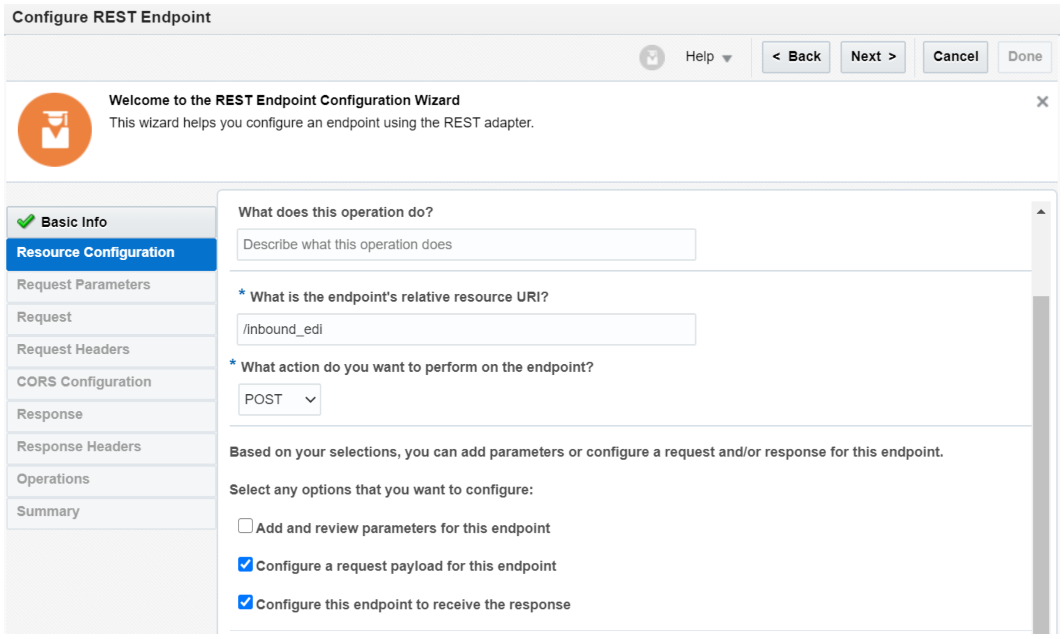
Configure the REST Adapter Trigger Connection

On the integration canvas, click the start node and select **Sample REST Endpoint Interface** as the trigger connection.

The Configure REST Endpoint wizard opens.

1. On the Basic Info page, enter the following details:
   a. In the **What do you want to call your endpoint?** field, enter `Receive-EDI`.
   b. Enter an optional description for the endpoint.
   c. Click **Next**.

2. On the Resource Configuration page:
   a. Enter `/inbound_edi` as the endpoint's relative resource URI.
   b. Select **POST** as the action to perform on the endpoint.
   c. Select to configure both request and response for this endpoint. Click **Next**.



Welcome to the REST Endpoint Configuration Wizard
This wizard helps you configure an endpoint using the REST adapter.

What does this operation do?
Describe what this operation does

* What is the endpoint's relative resource URI?
/inbound_edi

* What action do you want to perform on the endpoint?
POST

Based on your selections, you can add parameters or configure a request and/or response for this endpoint.

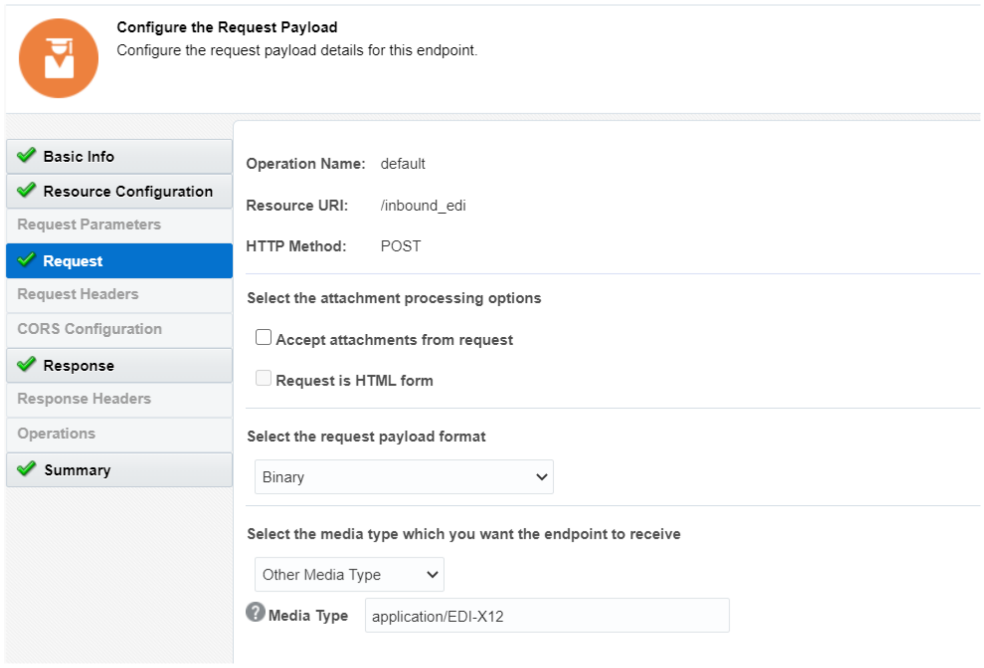
Select any options that you want to configure:

☐ Add and review parameters for this endpoint
☑ Configure a request payload for this endpoint
☑ Configure this endpoint to receive the response

3. On the Request page:
   a. Select **Binary** in the **Select the request payload format** field.

   > **Note:**
   >
   > Most EDI documents contain only textual data, but they may also include line breaks and special characters, which are used as delimiters. A few EDI documents contain raw binary data, such as images, along with text. Therefore, select the request payload format as **Binary**.

   b. Select **Other Media Type** as the media type you want the endpoint to receive.
   c. In the **Media Type** field, enter `application/EDI-X12`. Click **Next**.



Configure the Request Payload
Configure the request payload details for this endpoint.

Operation Name: default
Resource URI: /inbound_edi
HTTP Method: POST

Select the attachment processing options
☐ Accept attachments from request
☐ Request is HTML form

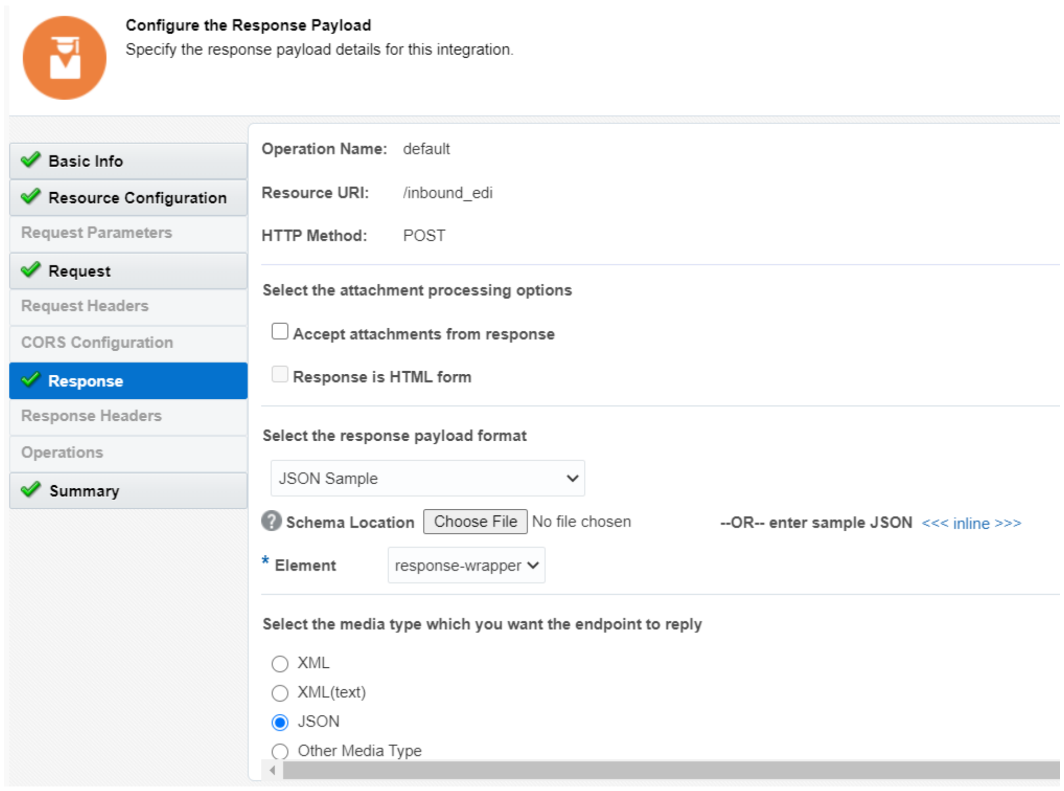
Select the request payload format
Binary

Select the media type which you want the endpoint to receive
Other Media Type
Media Type: application/EDI-X12

4. On the Response page:
   a. Select **JSON Sample** in the **Select the response payload format** field.
   b. Click the **inline** link next to **enter sample JSON**.
   c. In the resulting page, enter the following sample JSON and click **OK**.

```
{
   "translate_result": "xxxx",
   "hasError": false,
   "validationErrors": "xxxx",   "translated_payload":  "xxxx"
}
```

   d. Notice that **JSON** is automatically selected as the response media type.


Configure the Response Payload
Specify the response payload details for this integration.

Operation Name: default
Resource URI: /inbound_edi
HTTP Method: POST

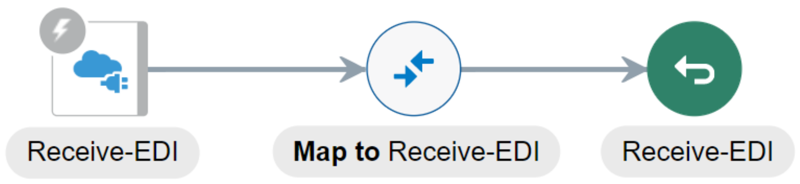
Select the attachment processing options
☐ Accept attachments from response
☐ Response is HTML form

Select the response payload format
JSON Sample
Schema Location Choose File No file chosen --OR-- enter sample JSON <<< inline >>>
* Element response-wrapper

Select the media type which you want the endpoint to reply
○ XML
○ XML(text)
● JSON
○ Other Media Type

5. Click **Next**, and on the **Summary** page, click **Done** to complete the REST Adapter configuration.

The integration flow is now represented as follows in the canvas:



Configure the B2B Action

Add a B2B action to the flow to translate an EDI document into an Oracle Integration XML message.

1. On the right side of the canvas, click **Actions**

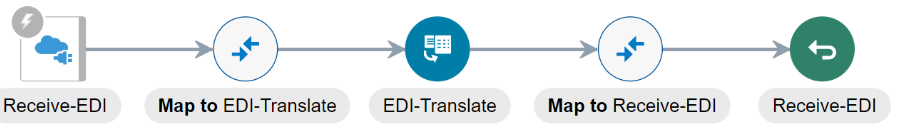
   , drag **EDI Translate**, and drop it after the first **Receive-EDI** element.

   The Configure B2B Action wizard opens.
2. On the Basic Info page, enter `EDI-Translate` as the name for the action, and click **Next**.
3. On the Select Data Formats page, enter the following details:
   a. Leave the **Inbound EDI message to Oracle Integration message** radio button selected.
   b. Select the document standard as **X12**.

c. Select the document version as **4010**.
d. Select the document type as **850 (Purchase Order).**
e. Select the document definition as **Standard**.
f. Select the EDI character encoding as **UTF8**.
g. Select **Yes** in the **Perform validations on input data?** field. Click **Next**.

4. On the Summary page, click **Done** to complete the configuration.

   Note that the corresponding mapping element is automatically added to the integration flow:

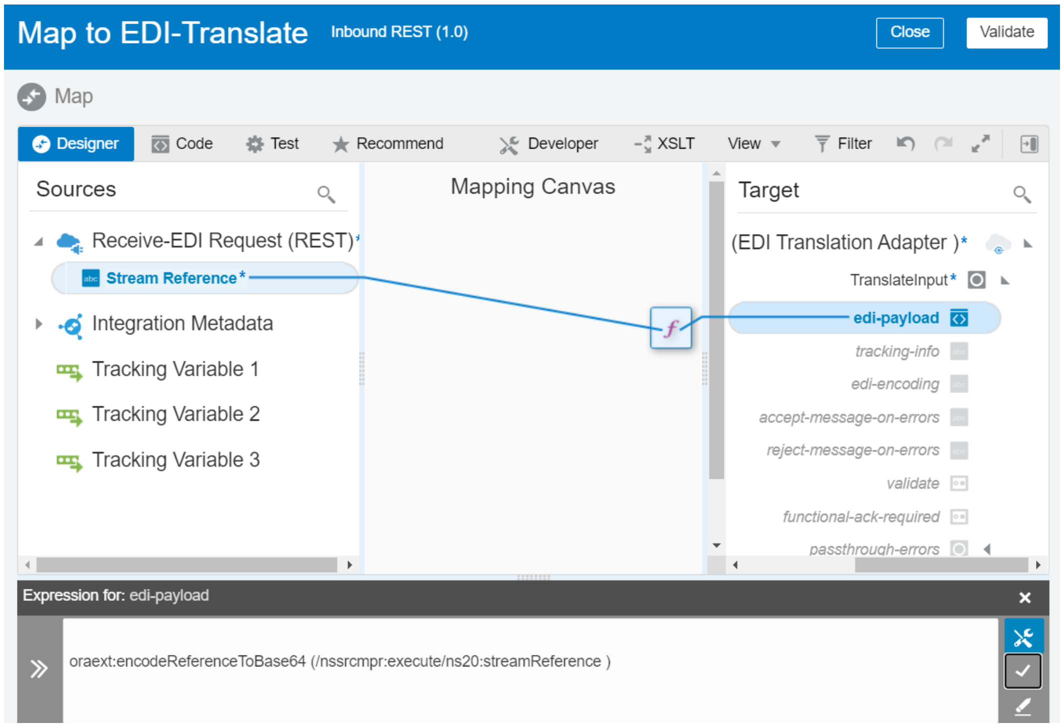


Configure Mapping Actions

Configure data mappings for the EDI-Translate action and Receive-EDI action in order to successfully parse the incoming EDI message and translate it to an XML message.

Configure the Map to EDI-Translate Action

1. Click the **Map to EDI-Translate** action and select **Edit**.
2. Map `streamReference` on the left to `edi-payload` on the right, within `TranslateInput`.
3. Click **Switch to Developer View** in the expression pane that appears at the bottom, and edit the expression for the mapping.
   Prefix the function call `encodeReferenceToBase64` to the existing edi-payload expression as follows: `oraext:encodeReferenceToBase64 (/nssrcmpr:execute/ns20:streamReference )`. Click **Save**.



```
oraext:encodeReferenceToBase64 (/nssrcmpr:execute/ns20:streamReference )
```

Note:

Your namespace may be different than `ns20`.

4. Click **Validate** and then **Close**.

Configure the Map to Receive-EDI Action

1. Click the **Map to Receive-EDI** action and select **Edit**.
2. Click **Developer** on the toolbar.
3. Map `edi-xml-document` (present on the left, under `$EDI-Translate` > `executeResponse` > `TranslateOutput`) to `translated_payload`, within the `response-wrapper` on the right.

Note:

To achieve this mapping, initially map the `@format` variable (within `edi-xml-document`) to `translated_payload`. In the expression for this mapping, delete `/ @format` and click **Toggle** on the toolbar. In the Components pane, expand **String** under **Functions**. Select the **get-content-as-string** function and drop it into the expression editor. Copy the existing expression in the editor into the function as follows: `oraext:get-content-as-string ($EDI-Translate/ nsmpr0:executeResponse/ns20:TranslateOutput/ns20:edi-xml-document )`.
Click **Save** ✓.

4. Map `translation-status` within the same `TranslateOutput` element to `translate_result` on the right.

5. Map `validation-errors-present` to `hasError`.
6. Map `validation-error-report` (`TranslateOutput` > `validation-errors`) to `validationErrors`.
7. Click **Validate** and then **Close**.

Activate the Integration

Check for errors, save, and activate the integration flow.

1. You'll notice an error notification on the canvas. To resolve it, click **Actions Menu** ☰ in the top-right corner of canvas, and select **Tracking**.
2. In the resulting dialog, select `streamReference` on the left and move it to the table on the right.
3. Click **Save**.
4. Save the integration and click **Close**.
5. On the Integrations page, click the **Activate** ⏻ button against your integration.
6. Click **Activate** in the Activate Integration dialog.

Parse Your First EDI Document

To execute your sample integration, send a request from a REST client tool, such as Postman.

Create a request definition on the REST client tool as follows:

1. Define the request to send an `HTTP POST` to the URL:

   ```
   http://host:port/ic/api/integration/v1/flows/rest/
   INBOUND_EDI_VIA_REST/1.0/inbound_edi
   ```

2. Configure authorization for the request. Select HTTP Basic Auth and provide the username and password to your Oracle Integration instance.
3. Add an HTTP header with `Content-Type` as **KEY** and `application/EDI-X12` as **VALUE**.
4. Use the sample EDI X12 purchase order provided in About Electronic Data Interchange as the request body. Copy the entire text of the EDI X12 document into the body field. If there are options for the type of request body, select **Raw** before you paste the EDI data.
5. Send the request from the REST client.
6. Verify the response from the integration flow.

   Check if the response status is 200 OK and the JSON is displayed as follows:

   ```
   {
     "translate_result": "Success",
     "hasError": false,
     "validationErrors": "",
     "translated_payload": "<edi-xml-document...>"
   }
   ```

The `translate_result` parameter with the value `Success` indicates that the EDI X12 document was parsed and translated successfully.

Test Syntactical Validations

In addition to translation, the B2B action also validates the EDI data it receives and reports the errors found.

As an integration developer, when you build integrations to send EDI X12 documents to backend applications for further processing, you can check for validation errors while parsing the EDI data and handle these errors appropriately in the integration flow. For details on error handling, see Parse and Transform an Inbound EDI Message. To enable input data validations, select **Yes** in the **Perform validations on input data?** field while configuring the B2B action. See Use the B2B Action in Standalone Mode.

To test the validation feature of the B2B action, let's introduce a syntactical error in the EDI X12 purchase order document and send it to the example B2B flow through the REST client.

1. In your REST client, in line 5 of the request body (containing the EDI X12 PO data), replace `USD` with `USD123` to make this data longer than the field allows.
2. Send the updated request from the client.
3. You'll see a response as follows:

```
{
  "translate_result": "Error",
  "hasError": true,
  "validationErrors": "[1] Error code: B2B-01752 | category: data_size |
message: (Severe) Element CUR02 (element id 100) has a identifier value
[USD123] of length 6, which is
  more than the maximum allowed length of 3. | segment-CUR > CUR02
(element id 100) | segment position 3 (starting with ST segment) |
element position 2 | line 4 |
  character position 7\n\n",
  "translated_payload": ...
}
```

The `translate_result` is now returned as `Error` and the syntactical error in the data is reported in the `validationErrors` element. This means the EDI X12 document was not successfully processed.

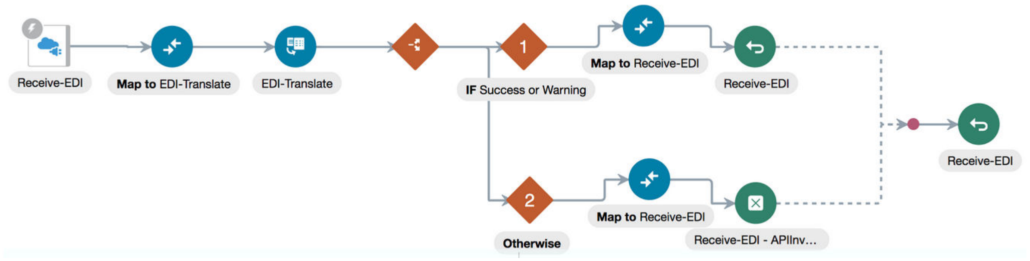
Parse and Transform an Inbound EDI Message

You can parse and transform an EDI message to match the XML format that a backend application can understand. The previous sections in this chapter described how to create a basic inbound integration using the B2B action. This section describes how to modify that integration to include a transformation step.

The previous integration returned the EDI-XML content directly in the REST Adapter response. This use case changes the REST Adapter response schema to return a FusionPurchaseOrder, a mock backend application schema for a purchase order. Because there is no real backend application to contact, the XML message is returned to the caller. To make this integration more useful, you can modify it further to add an adapter invoke connection that sends the transformed message to a backend application. Once you do that, you have an end-to-end purchase order flow.

Error handling is also introduced in the integration. When an EDI translation fails, an error is returned instead of transforming it to a backend message.

This use case is described in high-level steps below. It handles an input containing a single EDI document. To handle an EDI batch file, see Process an Inbound EDI Batch File Using the Stage File Action.



Perform the following changes to the integration created in the previous section:

1. Change the response schema in the Receive-EDI action (the REST Adapter trigger) created in Configure the REST Adapter Trigger Connection.
   a. Click the Receive-EDI element and select **Edit**.
   b. Go to the **Response** tab.
   c. Select **XML Schema** as the response payload format and **XML** as the media type.
   d. For the **Schema Location**, upload an `.xsd` file with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/PurchaseOrder"
xmlns:tns="http://www.example.org/PurchaseOrder"
elementFormDefault="qualified">
    <element name="FusionPurchaseOrder"
        type="tns:FusionPurchaseOrderType">
    </element>
    <complexType name="FusionPurchaseOrderType">
        <sequence>
            <element name="SenderId" type="string"></element>
            <element name="orderNumber" type="string"></element>
            <element name="orderDate" type="dateTime"></element>
            <element name="quoteID" type="string"></element>
            <element name="totalAmount" type="float"></element>
            <element name="currencyCode" type="string"></element>
            <element name="currencyConversionRate"
type="float"></element>
            <element name="lineItems" type="tns:OrderLineItem"
maxOccurs="unbounded" minOccurs="1"></element>
            <element name="billingAddress" type="tns:Address"></
element>
            <element name="shippingAddress" type="tns:Address"></
element>
            <element name="contact" type="tns:Contact"></element>
        </sequence>
    </complexType>
    <complexType name="OrderLineItem">
        <sequence>
            <element name="SKU" type="string"></element>
            <element name="Quantity" type="float"></element>
            <element name="unitOfMeasure" type="string"></element>
            <element name="comments" type="string"></element>
            <element name="price" type="float"></element>
            <element name="amount" type="float"></element>
        </sequence>
    </complexType>
    <complexType name="Address">
        <sequence>
            <element name="Name" type="string"></element>
            <element name="AddressLine1" type="string"></element>
            <element name="AddressLine2" type="string"></element>
            <element name="AddressLine3" type="string"></element>
            <element name="City" type="string"></element>
            <element name="State" type="string"></element>
            <element name="Country" type="string"></element>
            <element name="ZipCode" type="string"></element>
        </sequence>
    </complexType>
    <complexType name="Contact">
        <sequence>
            <element name="name" type="string"></element>
            <element name="phone" type="string"></element>
            <element name="email" type="string"></element>
        </sequence>
    </complexType>
</schema>
```

2. Add a switch action after the EDI-Translate action.
    a. For the If branch, enter the following expression:

    ```
    $EDI-Translate/nssrcmpr:executeResponse/nsmpr6:TranslateOutput/
    nsmpr6:translation-status = "Success" or $EDI-Translate/
    nssrcmpr:executeResponse/nsmpr6:TranslateOutput/nsmpr6:translation-
    status = "Warning"
    ```

    Note:

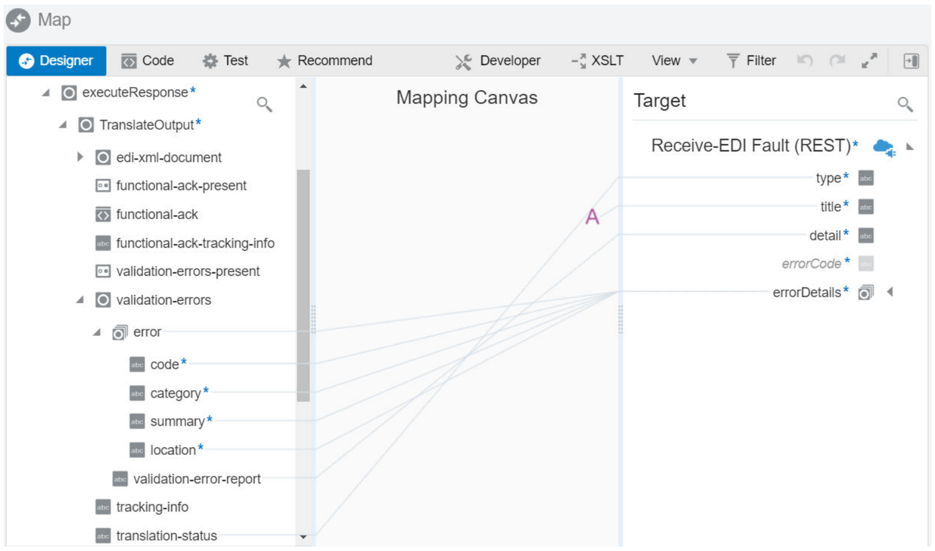
    Your namespace prefix may include different values than `nssrcmpr` and `nsmpr6`.

    This expression indicates that if `TranslateOutput > translation-status` has a value of `Success` or `Warning`, then take this route. This is referred to as the *success* route.

    b. Call the Otherwise branch the *error route* for error handling.
3. In the error route, add a fault return action.

This creates a map before the fault return action.

- In the map, connect the useful error information inside `TranslateOutput` > `validation-errors` to the fault error details. See B2B Action Input and Output Schema Reference for a description of elements inside `TranslateOutput`.



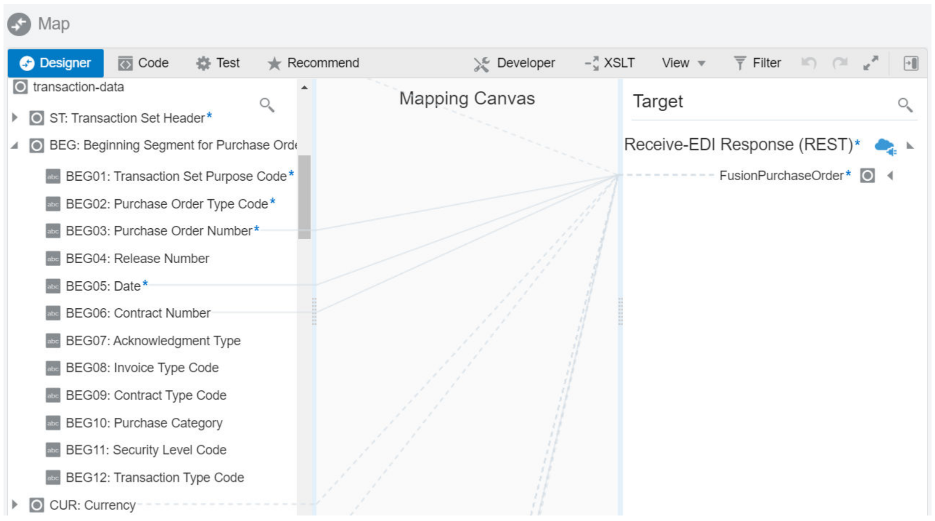
The error route does not transform the message (or prepare it for delivery to a backend application). Instead, it returns a 5*xx* error.

4. In the success route, add a message transformation step.

   Message transformation is done using a map action.

   - Add a map and a return action to this route.

     The map has fairly complex mappings from the EDI-XML purchase order to the FusionPurchaseOrder. Performing this mapping requires some expertise on the target application schema and EDI X12 schema.


Map
Designer
Code
Test
Recommend
Developer
XSLT
View
Filter
transaction-data
ST: Transaction Set Header*
BEG: Beginning Segment for Purchase Orde
BEG01: Transaction Set Purpose Code*
BEG02: Purchase Order Type Code*
BEG03: Purchase Order Number*
BEG04: Release Number
BEG05: Date*
BEG06: Contract Number
BEG07: Acknowledgment Type
BEG08: Invoice Type Code
BEG09: Contract Type Code
BEG10: Purchase Category
BEG11: Security Level Code
BEG12: Transaction Type Code
CUR: Currency
Mapping Canvas
Target
Receive-EDI Response (REST)*
FusionPurchaseOrder*

Elements are mapped from the X12 BEG and REF segments, purchase order line items from the PO1 loop, and the purchase order summary from CTT and AMT segments near the end.

Not all elements from the EDI-XML schema are mapped because the FusionPurchaseOrder schema is smaller and comparatively much simpler than the rich structure defined in the EDI X12 schema. The map transforms the EDI-XML message to a FusionPurchaseOrder message that is ready to be consumed directly by a back-end application.

5. Activate the integration and use any REST client such as `postman` to send the following EDI X12 850 purchase order:

```
ISA|00|          |00|          |01|111111T        |01|22222          |
190312|0845|U|00401|000001894|0|T|~
GS|PO|0124578|ACME|20190312|084515|0123456|X|004010
ST|850|1234
BEG|00|NE|PO-4503000||20190312|QO-1032
CUR|2L|USD|1.0000
REF|IT|999|Global Chips
REF|WO|P8923.5
REF|KY||Standard Terms and conditions will apply
PER|BD|JAMES SMITH|TE|1112223333|EM|jamess@globalchips.com
FOB|PC|OR|ORIGIN FREIGHT COLLECT|02|FOB
ITD||3|||||10|||||P03R
DTM|002|20190329
TD5|B||05||Ground
N1|BT|Global Chips
N2|ACCOUNTS PAYABLE
N3|P.O. BOX 1111
N4|NEW YORK|NY|10001|US
PO1|00001|1|EA|74.99|PE|BP|5566|VN|AB-1264
PID|F||||AB-1264 BRACKET ASSY WITH SPRING
PO1|00002|1|EA|25.00|PE|BP|7264|VN|DE-1834
PID|F||||DE-1834 GEAR BOX PACKAGE
CTT|2|2
AMT|TT|99.99
SE|22|1234
GE|1|0123456
IEA|1|000001894
```

You receive a response with the following XML message:

```
<tns:FusionPurchaseOrder xmlns:tns="http://www.example.org/PurchaseOrder">
    <tns:SenderId>111111T</tns:SenderId>
    <tns:orderNumber>PO-4503000</tns:orderNumber>
    <tns:orderDate>20190312</tns:orderDate>
    <tns:quoteID>QO-1032</tns:quoteID>
    <tns:totalAmount>99.99</tns:totalAmount>
    <tns:currencyCode>USD</tns:currencyCode>
    <tns:currencyConversionRate>1.0000</tns:currencyConversionRate>
    <tns:lineItems>
        <tns:SKU>AB-1264</tns:SKU>
        <tns:Quantity>1</tns:Quantity>
        <tns:unitOfMeasure>EA</tns:unitOfMeasure>
        <tns:comments>AB-1264 BRACKET ASSY WITH SPRING</
tns:comments>
        <tns:price>74.99</tns:price>
    </tns:lineItems>
    <tns:lineItems>
        <tns:SKU>DE-1834</tns:SKU>
        <tns:Quantity>1</tns:Quantity>
        <tns:unitOfMeasure>EA</tns:unitOfMeasure>
        <tns:comments>DE-1834 GEAR BOX PACKAGE</tns:comments>
        <tns:price>25.00</tns:price>
    </tns:lineItems>
    <tns:billingAddress>
        <tns:Name>Global Chips</tns:Name>
        <tns:AddressLine1>ACCOUNTS PAYABLE</tns:AddressLine1>
        <tns:AddressLine2>P.O. BOX 1111</tns:AddressLine2>
        <tns:AddressLine3/>
        <tns:City>NEW YORK</tns:City>
        <tns:State>NY</tns:State>
        <tns:Country>US</tns:Country>
        <tns:ZipCode>10001</tns:ZipCode>
    </tns:billingAddress>
    <tns:contact>
        <tns:name>JAMES SMITH</tns:name>
        <tns:phone>1112223333</tns:phone>
    </tns:contact>
</tns:FusionPurchaseOrder>
```

6. To test a negative case, modify line 1 and replace `ISA` with any other characters, and submit. This causes the integration to take the error route. An HTTP `Error 500` is returned with an error message, including this inner error:

   ```
   B2B-01858: Not an EDI X12 native document since the first 3 characters
   are not 'ISA' (found: 'SDS')
   ```

In conclusion, you can use Oracle Integration to accept an inbound EDI message, parse it, check for syntax errors in the EDI, transform it, and send it to a backend application.

Process an Inbound EDI Batch File Using the Stage File Action

Learn how to create a B2B flow, using the stage file and B2B actions, to process an inbound EDI batch file and send a functional acknowledgment for the file back to the sender.

Topics:

- Prerequisites to Set Up the Integration
- Build an Integration to Process the EDI Batch File
- Configure a Stage File Action to Debatch the EDI File
- Process an EDI Batch File with Heterogeneous Documents
- Send an EDI X12 997 Functional Acknowledgment

Prerequisites to Set Up the Integration

Review the prerequisites to set up a B2B flow in Oracle Integration to handle an inbound EDI batch file.

Before you begin configuring this example flow, you must:

- Have an external SFTP server to host the inbound EDI documents and the outbound functional acknowledgment (997) documents. Create an FTP connection to connect to the SFTP server. See Create an FTP Adapter Connection.
- Be familiar with creating, configuring, and activating integration flows in Oracle Integration. For a step-by-step guide to set up a basic B2B flow, see Create a Simple Integration Using B2B for Oracle Integration.
- Be familiar with the FTP invoke operation and the general usage pattern of the stage file action in integrations. See Process Files in Scheduled Integrations with a Stage File Action.

Note:

This example use case provides only a general guideline to implement the EDI batch file processing. If you do not have an external SFTP server, you can use any other adapter in place of the FTP Adapter to fetch the EDI file, and use any actions or constructs to build an integration to suit your use case. For instance, you can use the `streamReference` input available in the REST Adapter trigger connection of the Parse and Transform an Inbound EDI Message example as the input file. Subsequently, you can apply the stage file action and other processing steps on that file.

Build an Integration to Process the EDI Batch File

Follow the general guidelines provided here to build the example integration flow.

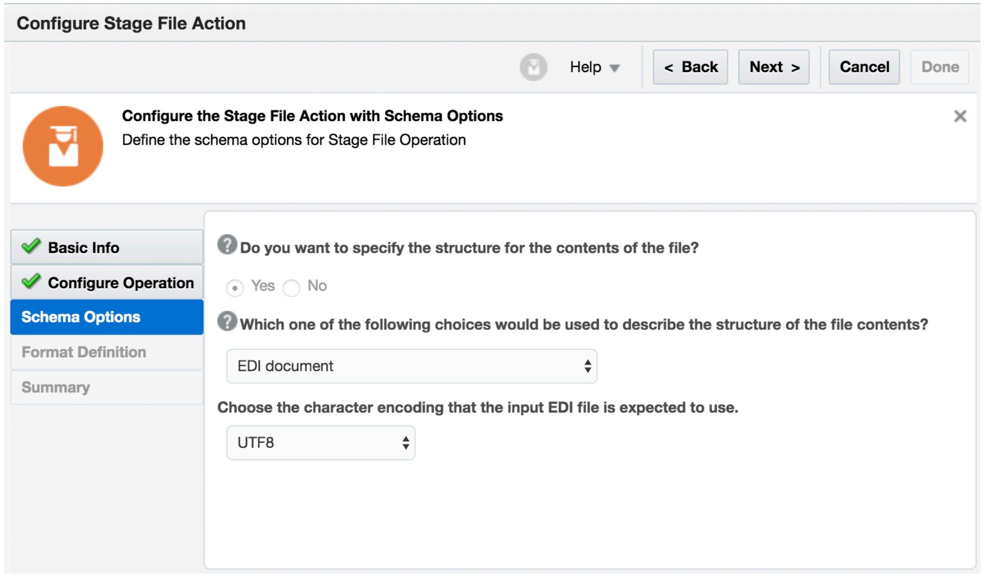
1. Download the incoming EDI batch file to Oracle Integration.
   You can use an FTP Adapter invoke connection to download a single file. Use the two FTP Adapter invoke connection patterns if you want to process multiple files from a directory on a remote SFTP server, where one FTP Adapter invoke connection retrieves a list of files, and another downloads the files.
2. Use a stage file action to debatch EDI documents from the downloaded batch file. See Configure a Stage File Action to Debatch the EDI File.
   Let's assume you have received a batch of fifty EDI purchase orders in a single file. The stage file action splits the file into fifty individual EDI purchase orders and offers them as repeating elements.
3. Using a for-each action, iterate over the repeating elements to handle one EDI document at a time.
4. Use the B2B action and parse one EDI document at a time.
   As the for-each action iterates over each individual EDI document, the B2B action keeps track that the original batch contains fifty EDI purchase orders, and finally performs some additional processing for the last document in the batch.

5. After the entire batch is parsed, send the functional acknowledgment document back to the original sender. See Send an EDI X12 997 Functional Acknowledgment.
   The B2B action generates an EDI X12 997 (Functional Acknowledgment) for the last EDI document it handles from a batch. In this example of fifty documents, the B2B action processes the first forty-nine EDI purchase orders routinely but includes an EDI X12 997 (Functional Acknowledgment) document to the fiftieth EDI purchase order. This acknowledgment document contains the status information for the entire batch of fifty documents, and according to the EDI X12 standard, you should transmit this document back to the originating trading partner after the entire batch has been parsed. If the B2B action has generated the 997 document, write it to a separate file and transmit the file to the trading partner using the FTP Adapter invoke connection and write file operations.
6. Check if the B2B action completed successfully.
   You can do this by verifying if `translation-status` is either a `Success` or `Warning`. If the status is either of the two, use a map action to transform the EDI-XML into a backend application message. Finally, use an invoke connection from a suitable adapter and send the message to the backend application.
7. After a file is processed, rename it with the `.processed` file extension to ensure that the same file is not picked for processing again in a future execution.

Configure a Stage File Action to Debatch the EDI File

Here's a list of key tasks you need to perform to configure the stage file action in your integration.

1. While configuring the stage file archive, select **Read File in Segments** in the **Choose Stage File Operation** field, which is present on the configuration wizard's Configure Operation page.
2. On the Schema Options page, specify the structure of the file contents as **EDI document** as shown in the following image. Specifying this option results in debatching of the EDI batch file into individual documents.


Configure Stage File Action

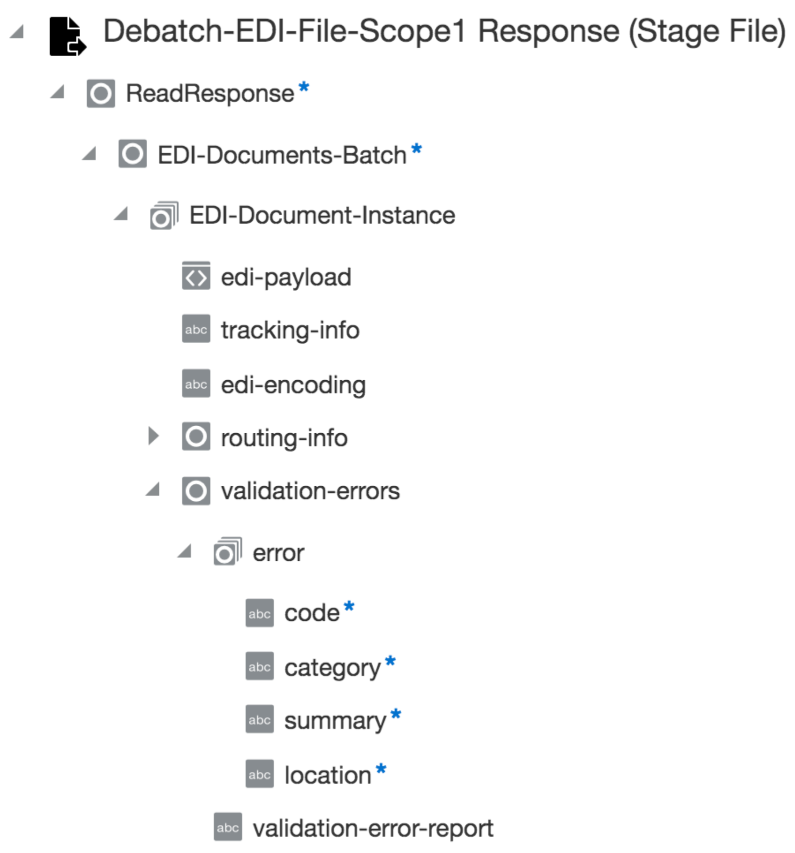
Configure the Stage File Action with Schema Options
Define the schema options for Stage File Operation

Do you want to specify the structure for the contents of the file?
[✓ Yes] No

Which one of the following choices would be used to describe the structure of the file contents?
EDI document

Choose the character encoding that the input EDI file is expected to use.
UTF8

3. The stage file action configured for EDI documents generates the following output structure:

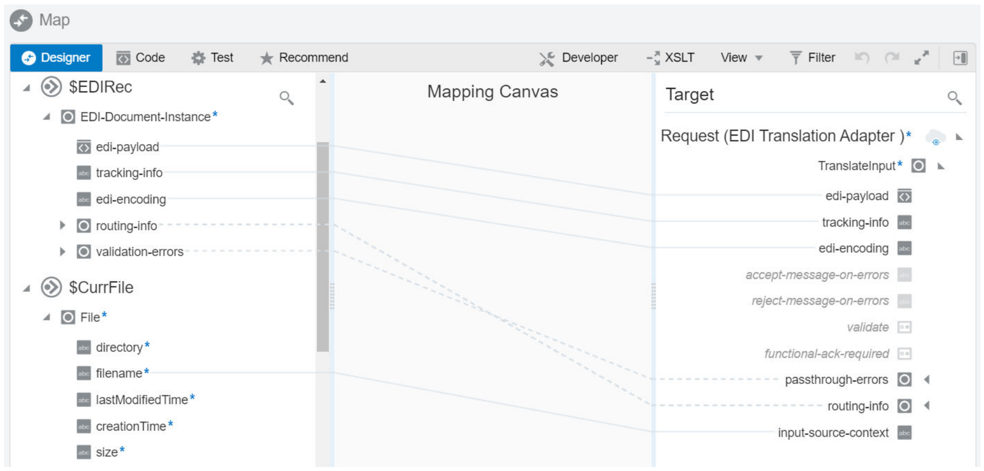


From the input batch of EDI documents, a repeating element **EDI-Document-Instance** is generated for each EDI document debatched. For example, if the input is a batch of fifty, there are fifty **EDI-Document-Instance** elements. If the input has one single EDI document, then only one **EDI-Document-Instance** element is generated.

Note:

In your integration, use a for-each action to iterate over the repeating element **EDI-Document-Instance**.

4. Configure the **Map to EDI-Translate** action.
   This action is the map that transfers data from the stage file action to the **EDI-Translate** action. The data includes the `edi-payload` as well as other elements required to track the state of an EDI batch; therefore, you must map the following elements:
   - `edi-payload`
   - `tracking-info`
   - `edi-encoding`
   - `routing-info` (map each child element from source to target)
   - `validation-errors` to `passthrough-errors` (map each child element from source to target)
   - (Optional) `input-source-context` from the target. See B2B Action Input and Output Schema Reference.



Note:

If your input file has a single EDI document (not a batch), the stage file action simply generates a single `EDI-Document-Instance` in the repeating element structure. In such a case, the for-each action iterates only once. An integration designed to handle an EDI batch, therefore, also works for single EDI document files. It is a special case that has a batch of one.

Error Handling

For each **EDI-Document-Instance**, the element **validation-errors** is populated with any errors detected during debatching. There are two important error categories:

- **EDI-Document-Instance > validation-errors > error > category = "FATAL"**
  This error category indicates that a critical error was found in the input EDI, causing the entire EDI file to be rejected (assuming the input file contained one X12 interchange). does not generate an X12 TA1 document to automatically report this error to the trading partner. Therefore, you must notify an administrator and work with the sender trading partner to have them correct and resend the file. A message with fatal errors must not be forwarded to a B2B action.

  Example error scenarios: The interchange/group control number is missing or the interchange control numbers in the ISA and IEA segments are not identical.

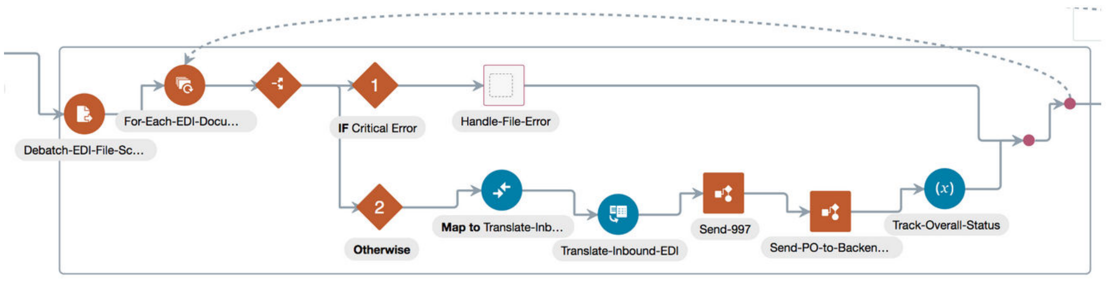
- **EDI-Document-Instance > validation-errors > error > category = "GROUP"**
  This error category indicates that an X12 functional group level error, which caused the entire X12 Group to be rejected. For this type of error, simply forward these errors to the next B2B action so that it can generate a proper EDI X12 997 Functional Acknowledgment that indicates the group was rejected. This is done with the map action explained later.

  Example error scenarios: If the group control numbers in the GS and GE segments are not identical.

Note:

The structural errors above are uncommon because the EDI is typically computer-generated. Nevertheless, it is good practice to incorporate error handling in your integration.

The following portion of an integration shows a stage file action named **Debatch-EDI-File-Sc**, a for-each action, a switch action with a route that checks for the presence of fatal errors, and another route that goes to a B2B action and other related actions:



The for-each action **For-Each-EDI-Docu** uses the iteration variable **$EDIRec**, in this example.

The integration routes an **EDI-Document-Instance** with fatal errors and sends it for error handing. When fatal errors are present, the B2B action must not be invoked.

The IF Critical Error route above uses the following logical expression to check for the presence of fatal errors:

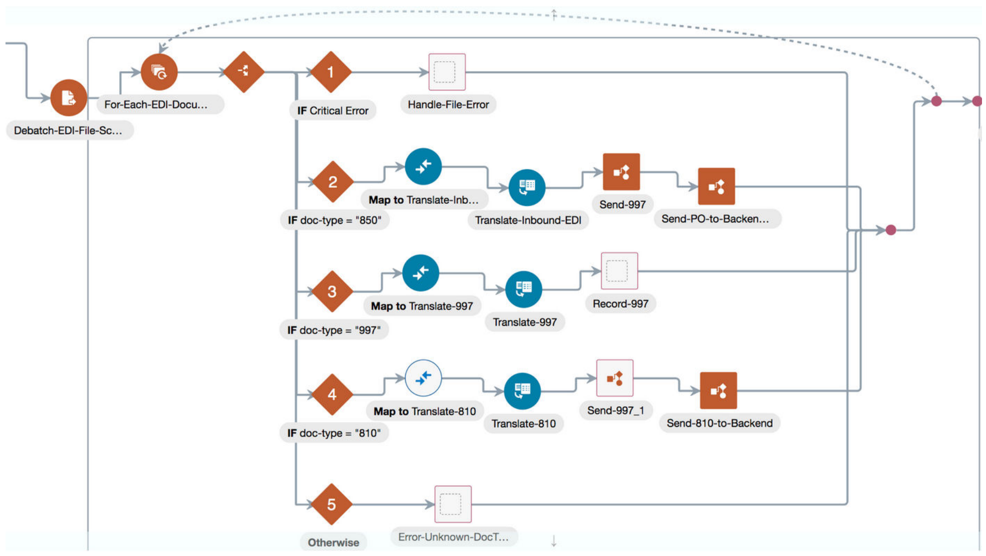
```
count($EDIRec > EDI-Document-Instance > validation-errors > error >
category[(text() = "FATAL")]) > 0
```

Process an EDI Batch File with Heterogeneous Documents

The EDI X12 standard allows a single EDI batch file to contain multiple EDI documents of different types. For instance, a batch file may contain fifty purchase orders and ten invoices, making it a heterogeneous batch.

To handle an EDI batch file containing different document types in your integration:

1. Add a switch action after the for-each action **For-Each-EDI-Doc** in the flow depicted in Configure a Stage File Action to Debatch the EDI File.
2. In the switch action, add one or more routes based on the value of the `$EDIRec > EDI-Document-Instance > routing-info > document-type` element or other child elements inside the `routing-info` element.
3. In each route, add a separate B2B action **EDI-Translate**, which you then configure to parse a specific document type. Ensure that you correctly route EDI documents to the matching B2B actions. Now, each B2B action parses a different document type, and you can then add error handling, transformations, and other steps to follow the action.
   An example integration showing the routes to handle fatal errors and EDI X12 documents 850, 997, and 810 is shown below. Route 5 handles document types not explicitly routed previously as an error.



The stage file action generates the following elements, as child elements to the `routing-info` element, that can drive the switch routes in an integration.

Child Element	Description
doc-standard	The EDI standard identified for the document, such as X12.
doc-type	The EDI document type; for example: 850, which represents an EDI X12 purchase order.
doc-version	The EDI version; for example, EDI X12 version 4010.
doc-definition	Not used.
ic-sender-ID-qualifier	The interchange sender ID qualifier that appears in the interchange header (for X12, the value from the ISA05 element of the ISA segment).
ic-sender-ID	The interchange sender ID that appears in the interchange header (for X12, the value from the ISA06 element of the ISA segment).
ic-receiver-ID-qualifier	The interchange receiver ID qualifier that appears in the interchange header (for X12, the value from the ISA07 element of the ISA segment).
ic-receiver-ID	The interchange receiver ID that appears in the interchange header (for X12, the value from the ISA08 element of the ISA segment).
grp-identifier-code	The functional group identifier code that appears in the functional group header (for X12, the value from the GS01 element of the GS segment).
grp-application-sender-code	The functional group application sender's code that appears in the functional group header (for X12, the value from the GS02 element of the GS segment).
grp-application-receiver-code	The functional group application receiver's code that appears in the functional group header (for X12, the value from the GS03 element of the GS segment).

Send an EDI X12 997 Functional Acknowledgment

The B2B action generates an EDI X12 997 (Functional Acknowledgment) for each functional group (GS) in the EDI batch document and handles it from a batch (even if a batch contains only one document).

A batch is defined as one X12 functional group starting with GS and ending with GE segments. A functional group contains one or more EDI documents, each one enveloped between a pair of ST and SE segments. One 997 (Functional Acknowledgment) is generated for one functional group. An X12 interchange is a container for one or more functional groups. The stage file and B2B actions, when used together, understand these envelope structures and generate as many 997 Acks as the number of functional groups that occur inside an X12 interchange.

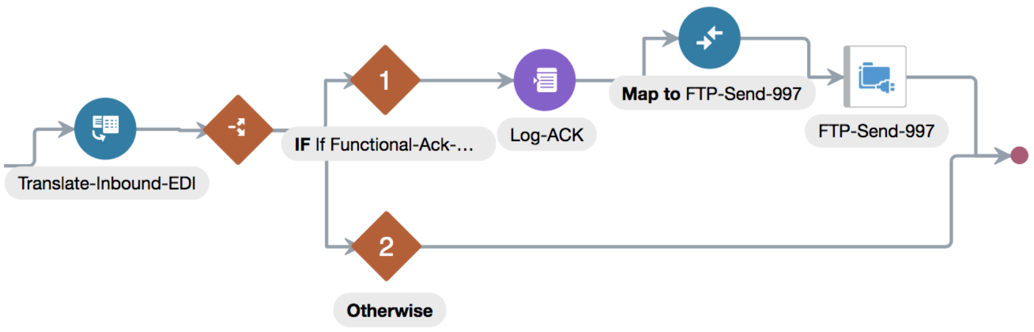
The following table depicts the output of a B2B action when it parses a batch of fifty EDI purchase orders. Each table row represents an individual EDI document that has been processed.

Note that only a few relevant fields from Elements in TranslateOutput for Inbound EDI are shown as the output here.

EDI Document Number (in a batch of 50)	**EDI Translate Output**
1	• `edi-xml-document` is populated. • `functional-ack-present` is `false`. • `functional-ack` is empty.
2	• `edi-xml-document` is populated. • `functional-ack-present` is `false`. • `functional-ack` is empty. .
...	...
49	• `edi-xml-document` is populated. • `functional-ack-present` is `false`. • `functional-ack` is empty.
50	• `edi-xml-document` is populated. • `functional-ack-present` is **`true`**. • `functional-ack` is **populated**.

In your integration, check for the `functional-ack-present` element. When its value is `true`, write the contents of the `functional-ack` element to a file (using a `decodeBase64ToReference` built-in function) and transmit the file to the originating trading partner using another FTP Adapter invoke connection. Note that the `functional-ack` element is populated as a Base64-encoded string. Therefore, you must apply a decode64 function or a variant of it in the map action to extract the EDI 997 Acknowledgment document.

The acknowledgment part of your integration resembles the following image:



Generate Outbound EDI from an Application Message

You can use Oracle Integration to create an EDI X12 document from a backend application XML message using the B2B action. The EDI X12 document is then sent to a trading partner.

This example describes the following aspects:

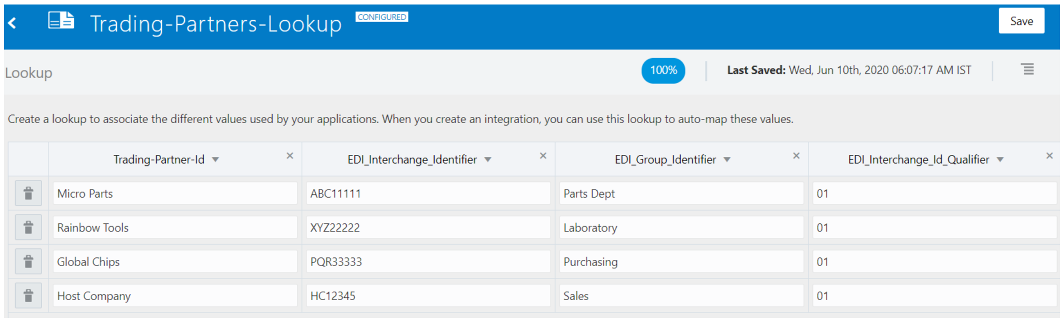
- Transforming a backend application XML message using the map action into an **edi-xml-document** that becomes an input to the B2B action in the outbound direction.
- Specifying delimiters of your choice to be used in the EDI.
- Specifying EDI identifiers using a lookup.
- Converting the **edi-xml-document** into an EDI X12 document and sending it to your trading partner.
- Using a separate integration to receive the EDI X12 997 Ack message and interpreting it.

Note:

In this use case, the backend application message is received as XML over REST. However Oracle Integration also supports JSON over REST. Alternatively you can use any other adapter, such as the Oracle ERP Cloud Adapter, to receive the message. This use case also provides only general guidelines such as usage of a lookup. You can also use alternate methods (such as retrieving the values by querying a database). Use any available action and adapter available in Oracle Integration to satisfy your use case.

Prerequisites

1. Navigate to **Integrations** > **Lookups**, and create a new lookup with the name `Trading-Partners-Lookup` as follows:



Last Saved: Wed, Jun 10th, 2020 06:07:17 AM IST

Create a lookup to associate the different values used by your applications. When you create an integration, you can use this lookup to auto-map these values.

Trading-Partner-Id	EDI_Interchange_Identifier	EDI_Group_Identifier	EDI_Interchange_Id_Qualifier
Micro Parts	ABC11111	Parts Dept	01
Rainbow Tools	XYZ22222	Laboratory	01
Global Chips	PQR33333	Purchasing	01
Host Company	HC12345	Sales	01

2. Add entries corresponding to the trading partners to which to send outbound EDI documents.
3. Add the following columns:
   - `Trading-Partner-Id`
   - `EDI_Interchange_Identifier`
   - `EDI_Group_Identifier`
   - `EDI_Interchange_Id_Qualifier`

   This lookup is used in the integration to fetch the EDI identifiers corresponding to a given trading partner.
4. Add an entry for the host company, so your company's EDI identifiers are also entered.

General Guidelines to Build an Integration for Outbound EDI Processing

1. Receive the backend application XML message using a REST Adapter trigger connection.
   One of the elements in the input XML must specify a trading partner that is the recipient of the outbound EDI. For example, the input XML has:

   ```
   <tradingPartnerId>Micro Parts</tradingPartnerId>
   ```

   In this case, the recipient trading partner is Micro Parts, which must have a matching entry in the lookup you created. The lookup allows the integration to retrieve EDI identifiers for Micro Parts dynamically.
2. Add the B2B action next and configure it for the outbound direction.
3. Select **X12** as the **Document Standard**, **4010** as the **Document Version**, and **850 (Purchase Order)** as the **Document Type**.
4. Configure the map action that occurs between the REST Adapter trigger connection and the B2B action to transform the backend application XML message to the **edi-xml-document**.
   Reference the lookup in the map expression to retrieve the EDI identifiers. This is explained later in this section.
5. Check if the EDI translate completed successfully by verifying that **translation-status** is either a `Success` or `Warning`.
   If successful, use an adapter invoke connection to send the EDI document to the trading partner. For example, use an FTP Adapter invoke connection to deliver the EDI document to an external B2B Value Added Network (VAN) provider to forward to a trading partner in a final push.

6. Create a separate integration using the inbound EDI pattern described in Process An Inbound EDI Batch File Using Stage File and Send a 997 Acknowledgment to receive the EDI X12 997 (Functional Acknowledgment) document type. This is explained in more detail later in this section.

Specifying EDI Delimiters of Your Choice

The B2B action enables you to specify EDI delimiters of your choice by assigning them to XML attributes inside the **edi-xml-document** > **headers** > **interchange-ctrl** element. If you don't set a value, a default value is used.

Assign a single ASCII character to any of the attributes. To specify special characters, nonprintable characters, or Unicode characters, see section Elements in TranslateInput for the supported values.

The following attributes represent each delimiter:

EDI X12 Delimiter	XML Attribute Where to Set a Value	Default Value
Element delimiter	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'element-separator'**	*
Segment terminator	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'segment-terminator'**	~
Composite element delimiter	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'subelement-separator'**	:
Repetition separator (only used in X12 4020 and newer versions)	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'repetition-separator'**	N/A

Note:

Ensure that data values do not contain any of the delimiter characters. Specify delimiters that do not occur in the data. Otherwise, the generated EDI cannot be parsed correctly. Each delimiter must also use a distinct character. Using the same character for any two delimiters generates an invalid EDI message.

Inserting EDI Identifiers in the Interchange and Group Headers

Outbound EDI X12 mandates the interchange and group envelope header segments (the ISA and GS segments) to specify EDI identifiers. These are mandatory elements that are not assigned default values. You must specify values for the six header fields described below, at a minimum. The remaining header fields are assigned default values.

The lookup created earlier is used to populate these values:

EDI X12 Header Field	Guidelines for Mapping Based on the Trading-Partner-Lookup	Target Element
ISA05: Interchange Sender ID Qualifier	This is a two-character value that must use a code from an X12 code list for this element. The expected value is the interchange ID qualifier of the sending party, which is your company (that is, the host company). From the lookup, fetch the **EDI_Interchange_Id_Qualifier** for the **Host Company** entry and assign that as the value. A sample mapping expression is: `dvm:lookupValue ("tenant/resources/dvms/Trading-Partners-Lookup", "Trading-Partner-Id", "Host Company", "EDI_Interchange_Id_Qualifier", "Trading-Partners-Lookup error" )`	**edi-xml-document headers** > **interchange-ctrl** > **attribute 'sender-id-qualifier'**
ISA06: Interchange Sender ID	The value is a maximum of 15 characters. The expected value is the interchange ID of the sending party, which is your company (that is, host company). From the lookup, fetch the **EDI_Interchange_Identifier** for the entry **Host Company** and assign that as the value. For example, the mapping expression is: `dvm:lookupValue ("tenant/resources/dvms/Trading-Partners-Lookup", "Trading-Partner-Id", "Host Company", "EDI_Interchange_Identifier", "Trading-Partners-Lookup error" )`	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'sender-id'**

EDI X12 Header Field	Guidelines for Mapping Based on the Trading-Partner-Lookup	Target Element
ISA07: Interchange Receiver ID Qualifier	This is a two-character value and must use a code from an X12 code list for this element. The expected value is the interchange ID qualifier of the receiving party, which is your remote trading partner. From the lookup, fetch the **EDI_Interchange_Id_Qualifier** for the entry specified by the input **tradingPartnerId** element, and assign that as the value. For example, the mapping expression is: `dvm:lookupValue ("tenant/resources/dvms/Trading-Partners-Lookup","Trading-Partner-Id", /nssrcmpr:execute/tns:AcmePurchaseOrder/tns:tradingPartnerId, "EDI_Interchange_Id_Qualifier", "Trading-Partners-Lookup error" )`	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'receiver-id-qualifier'**
ISA08: Interchange Receiver ID	The value is a maximum of 15 characters. The expected value is the interchange ID of the receiving party, which is your remote trading partner. From the lookup, fetch the **EDI_Interchange_Identifier** for the entry specified by the input **tradingPartnerId** element, and assign that as the value. For example, the mapping expression is: `dvm:lookupValue ("tenant/resources/dvms/Trading-Partners-Lookup", "Trading-Partner-Id",/nssrcmpr:execute/tns:AcmePurchaseOrder/tns:tradingPartnerId,"EDI_Interchange_Identifier", "Trading-Partners-Lookup error" )`	**edi-xml-document** > **headers** > **interchange-ctrl** > **attribute 'receiver-id'**

EDI X12 Header Field	Guidelines for Mapping Based on the Trading-Partner-Lookup	Target Element
GS02: Application Sender's Code	The value is between 2 and 15 characters. The expected value is the application sender's code of the sending party, which is your company (that is, host company). From the lookup, fetch the **EDI_Group_Identifier** for the entry **Host Company**, and assign that as the value. For example, the mapping expression is: `dvm:lookupValue ("tenant/resources/dvms/Trading-Partners-Lookup","Trading-Partner-Id", "Host Company", "EDI_Group_Identifier", "Trading-Partners-Lookup error" )`	**edi-xml-document** > **headers** > **group** > **app-senders-code**
GS03: Application Receiver's Code	The value is between 2 and 15 characters. The expected value is the application receiver's code of the receiving party, which is your remote trading partner. From the lookup, fetch the **EDI_Group_Identifier** for the entry specified by the input **tradingPartnerId** element, and assign that as the value. For example, the mapping expression is: `dvm:lookupValue ("tenant/resources/dvms/Trading-Partners-Lookup", "Trading-Partner-Id", /nssrcmpr:execute/tns:AcmePurchaseOrder/tns:tradingPartnerId, "EDI_Group_Identifier", "Trading-Partners-Lookup error" )`	**edi-xml-document** > **headers** > **group** > **app-receivers-code**

EDI Control Numbers Are Generated Automatically

An outbound EDI X12 document also mandates three control numbers, and the B2B action generates those automatically with unique incremented values.

- Interchange control number, populated in the ISA13 element and IEA02 element.
- Group control number, populated in the GS06 element and GE02 element.
- Transaction set control number, populated in the ST02 element and SE02 element.

Similarly, the **Date** and **Time** fields in the **Interchange** and **Group** headers are also automatically generated based on the current timestamp. Other remaining fields are assigned default values, including those in the **edi-xml-document** > **trailers** element.

You can override any of the default values by assigning them with your values explicitly to the **interchange-ctrl** and **group** attributes.

Checking For Translation Errors

You must check for validation errors that may be reported during EDI generation. Validation errors mean that the generated EDI is syntactically invalid and is likely rejected by your trading partner. You must resolve the errors by correcting the mappings or the input data.

Check for validation errors using a switch action and the success condition as **TranslateOutput** > **translation-status = "Success" or TranslateOutput** > **translation-status = "Warning"**. Otherwise, if **TranslateOutput** > **translation-status is 'Error'**, it indicates there are critical validation errors.

Output EDI Payload

If EDI translation completes successfully (with no errors or only warnings), the **TranslateOutput** > **edi-payload** element is populated with the generated EDI. This value is Base64-encoded and you must apply a decode64 function or variant to get the actual EDI document at the point of delivery to a trading partner (for example, while writing it to file to be sent through an FTP Adapter invoke connection).

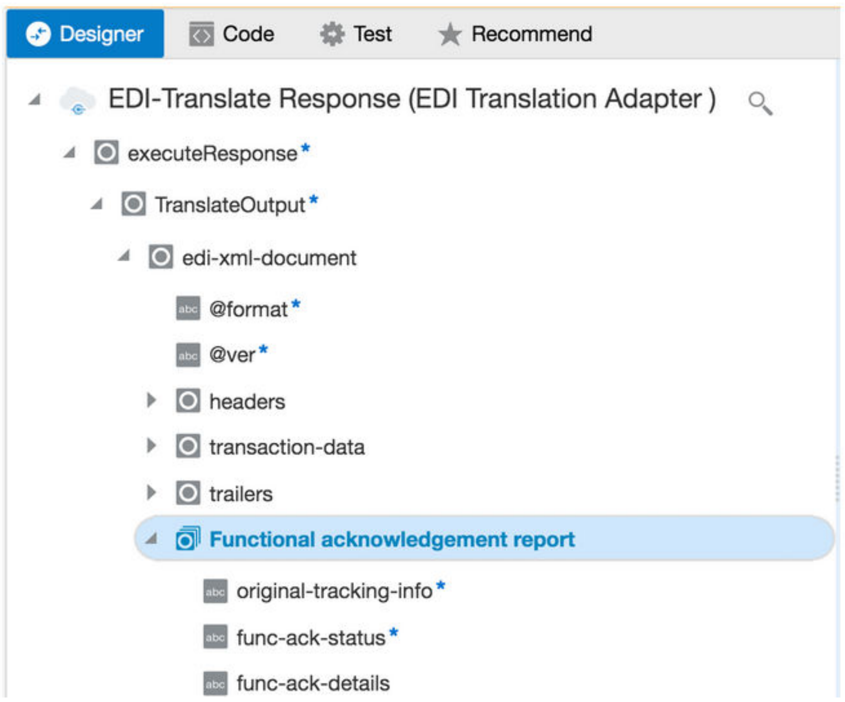
Receiving Inbound EDI X12 997 Functional Acknowledgments

When the remote trading partner receives your EDI document, an EDI X12 997 document is processed and sent back as an interim acknowledgment. An 997 Ack message reports a status, such as accepted or rejected, indicating whether your trading partner parsed the original EDI document you sent. When you receive a 997 Ack message, it is important to inspect it to determine the outcome reported by your trading partner. If rejected, you can take corrective actions.

General Guidelines to Build an Integration to Receive an 997 Ack

You must build a separate integration using the inbound EDI pattern described in Parse and Transform an EDI Message.

1. In the B2B action, select the **997 (Functional Acknowledgment)** document type.
2. Remove the transformation step, as this integration does not need to produce a backend application message. Backend systems typically do not need or support the B2B layer acknowledgments.
3. Add a check for the functional acknowledgment status field located at, **TranslateOutput** > **edi-xml-document** > **func-ack-report (same as Functional acknowledgment report)** > **func-ack-status**.



EDI-Translate Response (EDI Translation Adapter)
executeResponse*
TranslateOutput*
edi-xml-document
@format*
@ver*
headers
transaction-data
trailers
Functional acknowledgement report
original-tracking-info*
func-ack-status*
func-ack-details

This element has a value of **Success**, **Warning**, or **Error** to represent the acceptance status reported by your trading partner. In case of an error, the **func-ack-details** field is populated with plain text describing the interpretation of what the 997 Ack message says.

See the description for each of the child elements under **func-ack-report (same as Functional acknowledgment report)** in the chapter, B2B Action Input and Output Schema Reference.

Note:

The B2B action never generates another 997 Ack message when it parses an inbound 997 Ack document. The EDI X12 standard defines the behavior that there is no Ack for an Ack message. Otherwise, it implies a procedural infinite loop.

4

B2B Documents and B2B Schemas

Within orchestrated integrations, you can use B2B for Oracle Integration through the B2B action. When you add this action to an integration flow, the Configure B2B Action Wizard is invoked where you can specify all the details for the translate operation, including the operation direction (inbound or outbound), document standard, document definition to use, and so on.

Topics:

- Work with B2B Documents
- Work with B2B Schemas

Work with B2B Documents

B2B for Oracle Integration allows you to customize standard EDI documents by adding new schema constructs or by editing existing constructs.

On the Oracle Integration home page, click **B2B** on the left navigation pane, then select **B2B Documents** to access the B2B Documents page.

Topics:

- Create a Custom B2B Document Definition
- Edit or Clone a B2B Document

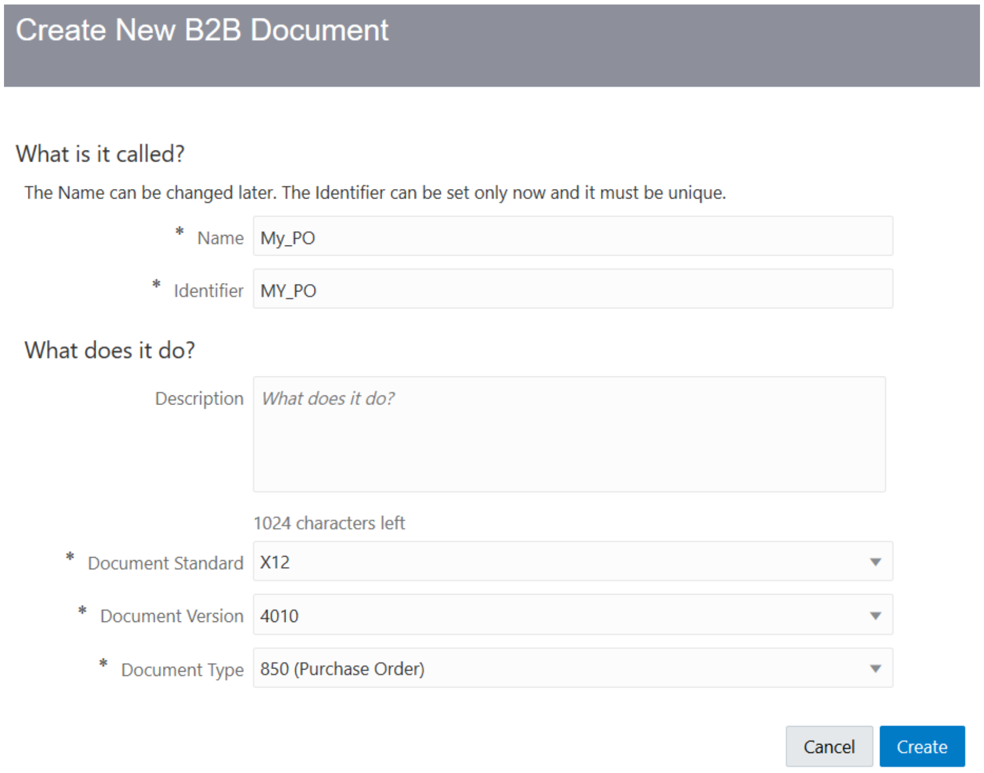
Create a Custom B2B Document Definition

You can create customized document definitions to use in your B2B integrations. Custom document definitions are useful for scenarios in which your trading partner requires specific customizations to meet certain business requirements.

1. On the B2B Documents page, click **Create**.
2. Enter the following details to create a new B2B document definition.

Element	Description
Name	Enter a document name.
Identifier	This field is automatically populated with the document name. You can manually change this value.
Description	Enter an optional description of the customization details for this document.
Document Standard	• EDIFACT • X12
Document Version	Select the document version. The versions shown are based on the document standard you selected.
Document Type	Select the document type.

For example:



Create New B2B Document

What is it called?

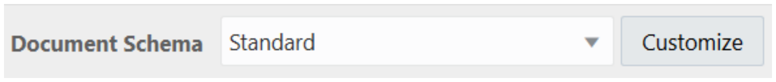
The Name can be changed later. The Identifier can be set only now and it must be unique.

* Name	My_PO
* Identifier	MY_PO

What does it do?

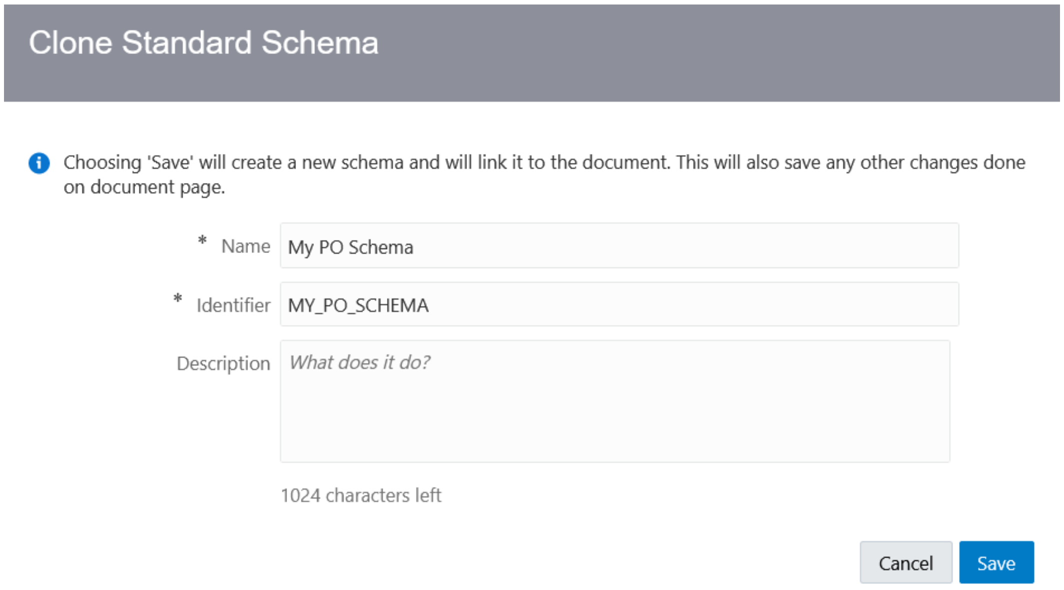
Description	*What does it do?*
	1024 characters left
* Document Standard	X12
* Document Version	4010
* Document Type	850 (Purchase Order)

3. Click **Create**.
   The details page for your new B2B document is displayed. The **Document Schema** field shows **Standard** as the schema type by default.



Document Schema: Standard

4. Click **Customize** to customize the standard schema to satisfy your business requirements. If you had previously created custom schemas, they are also displayed for selection in the dropdown list. If you want, you can select those schemas to create further customizations.
   The Clone Standard Schema dialog is displayed.
5. Enter a name and optional description for your custom schema.



Clone Standard Schema

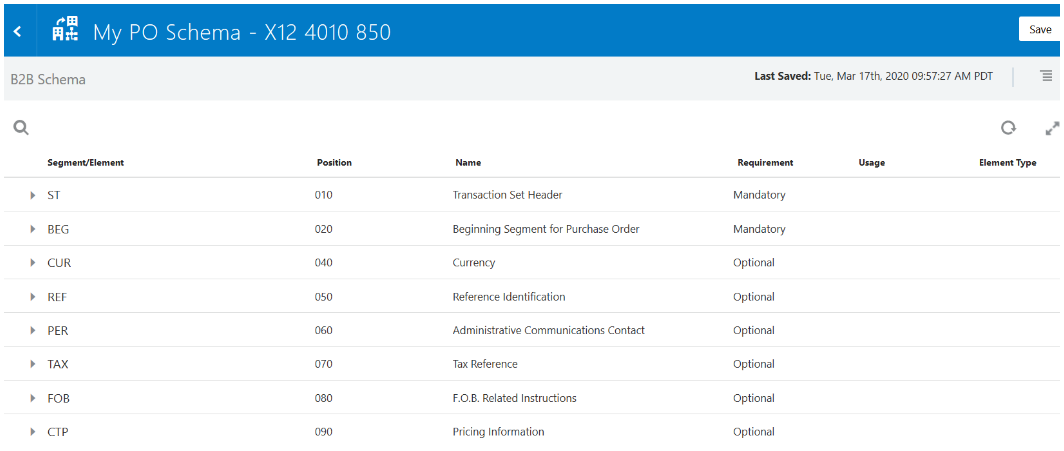
Choosing 'Save' will create a new schema and will link it to the document. This will also save any other changes done on document page.

* Name	My PO Schema
* Identifier	MY_PO_SCHEMA
Description	*What does it do?*

1024 characters left

Cancel Save

6. Click **Save**.
   This action creates a copy of the standard EDI X12 or EDIFACT schema for you to use as a baseline to customize. For this example, the EDI X12 schema is shown.

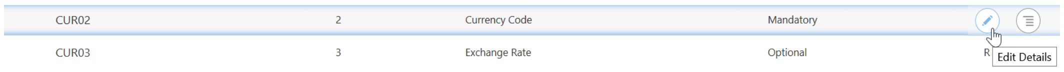


My PO Schema - X12 4010 850 Save

B2B Schema **Last Saved:** Tue, Mar 17th, 2020 09:57:27 AM PDT

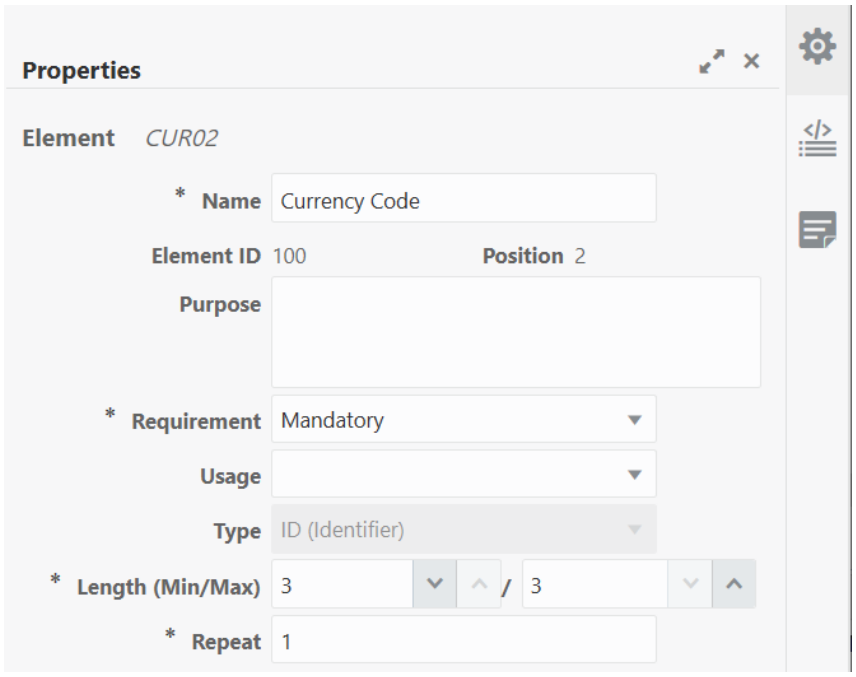
Segment/Element	Position	Name	Requirement	Usage	Element Type
ST	010	Transaction Set Header	Mandatory		
BEG	020	Beginning Segment for Purchase Order	Mandatory		
CUR	040	Currency	Optional		
REF	050	Reference Identification	Optional		
PER	060	Administrative Communications Contact	Optional		
TAX	070	Tax Reference	Optional		
FOB	080	F.O.B. Related Instructions	Optional		
CTP	090	Pricing Information	Optional		

7. On the schema page, find the element you want to customize, and select **Edit Details**. As an example, select the **CUR02** currency code element (part of the **CUR** segment) to edit it. You can also add new constructs to the schema. See Edit Properties of Segments and Other Schema Constructs.



CUR02	2	Currency Code	Mandatory
CUR03	3	Exchange Rate	Optional

The Details pane is displayed with the **Properties** tab selected.



Element CUR02

* Name: Currency Code

Element ID: 100 Position: 2

Purpose:

* Requirement: Mandatory

Usage:

Type: ID (Identifier)

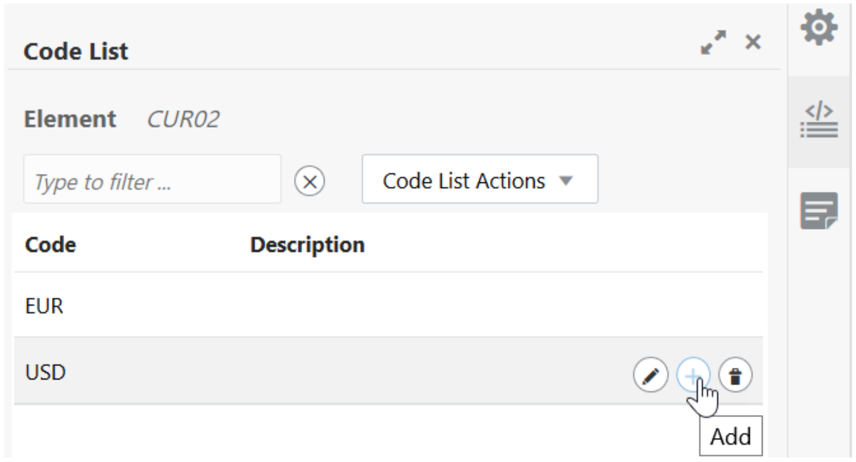
* Length (Min/Max): 3 / 3

* Repeat: 1

The three tabs on the Details pane enable you to perform customizations.

Tab	Description
Properties	Displayed by default when you initially access the Details pane. You can modify the following standard EDI X12 properties for the selected segment or element: • Purpose of element • Requirement (mandatory, optional, or conditional) • Usage (must use, do not use, recommended, or not recommended) • Minimum and maximum number of characters • Number of times to repeat the element
Code List	Click to view the code list for the element. A code list defines an enumeration of allowed values for the element.
Notes	Click to add notes to the element.

8. Edit the element details as necessary for your business environment. For this example, select **Code List** to add new code.
9. Click **Add a New Code List** (if the element does not already have a code list defined).
10. A placeholder code is created for you. Hover over it and select **Edit.** Add a currency code name and optional description, and click **x** to save your updates.
11. Click **Add** to add more codes and edit them as necessary for your business requirements. For this example, add `EUR` (for the Euro currency) and `USD` (for US dollars).

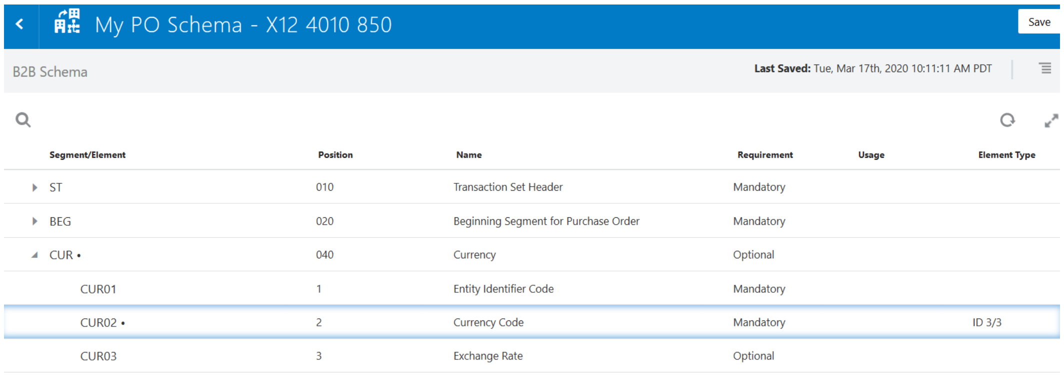


Code List

Element CUR02

Code	Description
EUR	
USD	

12. When complete, click **X** in the upper right corner to close the dialog. Click **Save**.
    The segment (**CUR**) and the element (**CUR02**) that you customized to deviate from the standard schema are identified by a dot.



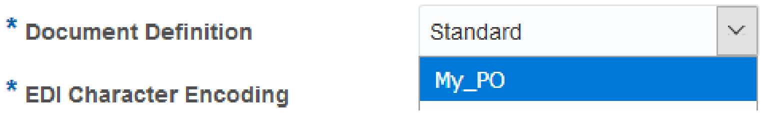
My PO Schema - X12 4010 850

B2B Schema

Last Saved: Tue, Mar 17th, 2020 10:11:11 AM PDT

Segment/Element	Position	Name	Requirement	Usage	Element Type
ST	010	Transaction Set Header	Mandatory		
BEG	020	Beginning Segment for Purchase Order	Mandatory		
CUR •	040	Currency	Optional		
CUR01	1	Entity Identifier Code	Mandatory		
CUR02 •	2	Currency Code	Mandatory		ID 3/3
CUR03	3	Exchange Rate	Optional		

13. Return to the B2B document's details page and click **Save** to associate the customized schema to the document.
14. Go to an orchestrated integration either to add a new B2B action or edit an already configured B2B action that is not part of an activated integration. See Use the B2B Action in Standalone Mode.
15. Note that the customized document definition is now available to use from the **Document Definition** list in the wizard.



* Document Definition: Standard

* EDI Character Encoding

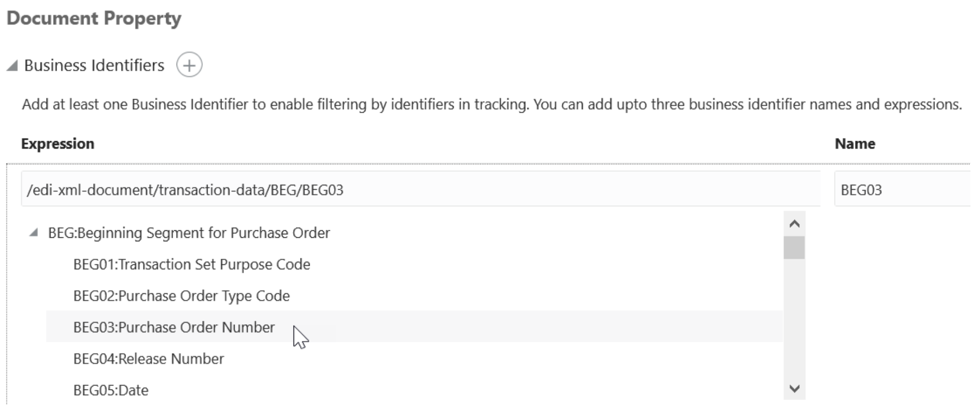
Edit or Clone a B2B Document

You can edit an existing B2B document to associate a different schema or add business identifiers. You can also clone a document to create a copy of it.

1. Hover over a document row to see the actions you can perform on a B2B document.
2. Click to change the document's name, description, or associated schema or add a business identifier.

> **Note:**
>
> You cannot change the document standard, version, or type.

3. Add a minimum of one and a maximum of three business identifier names and expressions to enable filtering by the identifiers on the Track Instances page when this document is sent or received. For example, you can search for a specific **Purchase Order Number** if you select **BEG03:Purchase Order Number** as the primary identifier.



Document Property

Business Identifiers

Add at least one Business Identifier to enable filtering by identifiers in tracking. You can add upto three business identifier names and expressions.

Expression	Name
/edi-xml-document/transaction-data/BEG/BEG03	BEG03

- BEG:Beginning Segment for Purchase Order
  - BEG01:Transaction Set Purpose Code
  - BEG02:Purchase Order Type Code
  - BEG03:Purchase Order Number
  - BEG04:Release Number
  - BEG05:Date

4. Click **Actions** to clone or delete a document.
5. Click **Open Details** to view the document details.

Work with B2B Schemas

B2B for Oracle Integration allows you to create a new schema using a standard schema or by importing a Standard Exchange Format (SEF) file. In addition, you can perform a host of customizations on the schemas and generate implementation guides for custom or standard schemas.

On the Oracle Integration home page, click **B2B** on the left navigation pane, then select **B2B Schemas** to access the B2B Schemas page.

Topics:

- Create a New B2B Schema
- Create a Schema from an Imported SEF File
- Customize or Edit a B2B Schema
- Generate an Implementation Guide

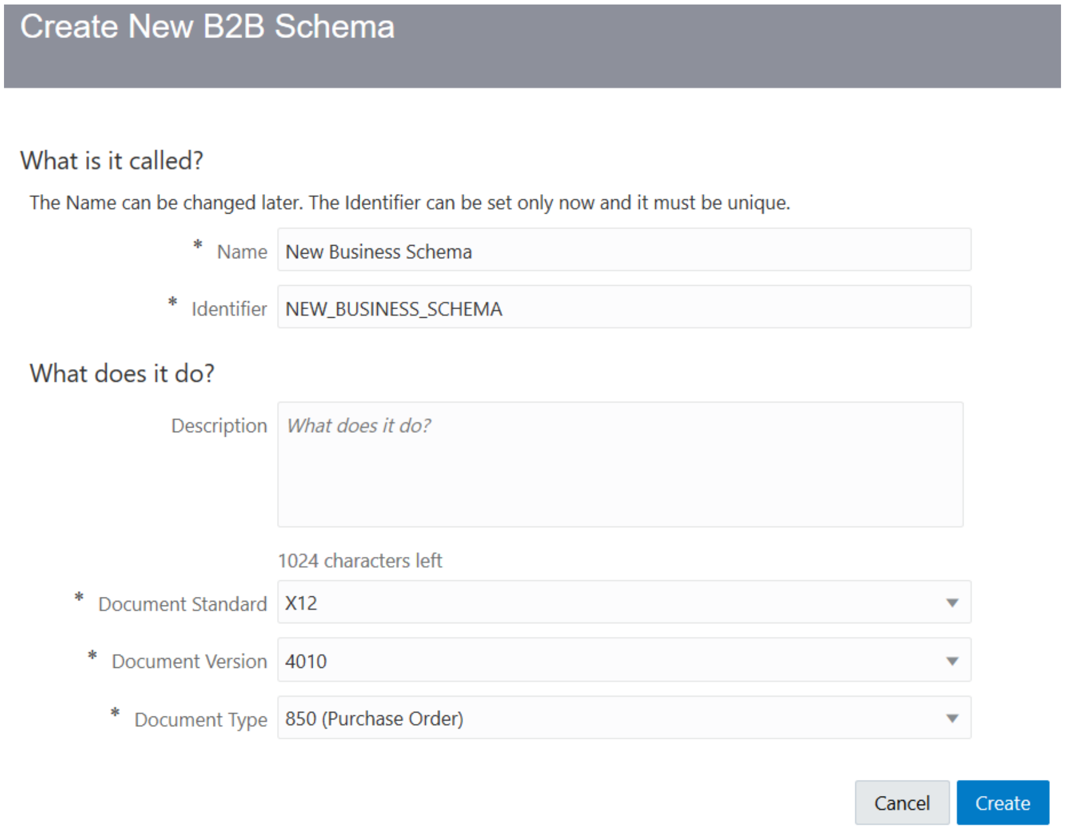
Create a New B2B Schema

You can create a new B2B schema based on a standard document type and customize it to match your requirements.

1. On the B2B Schemas page, click **Create**.

2. Enter the following details.

Element	Description
Name	Enter a document name.
Identifier	This field is automatically populated with the document name. You can manually change this value.
Description	Enter an optional description of the customization details for this document.
Document Standard	Select the document standard. The document standard identifies the business protocol to follow when exchanging business documents between partners. • EDIFACT • X12
Document Version	Select the document version.
Document Type	Select the document type.


Create New B2B Schema

What is it called?

The Name can be changed later. The Identifier can be set only now and it must be unique.

* Name	New Business Schema
* Identifier	NEW_BUSINESS_SCHEMA

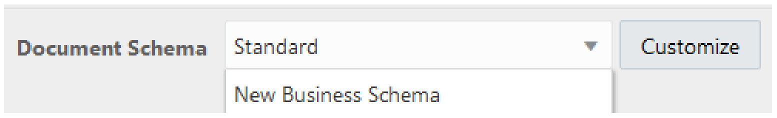
What does it do?

Description	*What does it do?*
	1024 characters left
* Document Standard	X12
* Document Version	4010
* Document Type	850 (Purchase Order)

3. Click **Create**.
4. Customize the schema to match your business requirements.

   For an example customization, see Step 7 of Create a Custom B2B Document Definition.

   After completing the customization, you can select the custom schema while creating a new B2B document as described in Step 3 of Create a Custom B2B Document Definition.

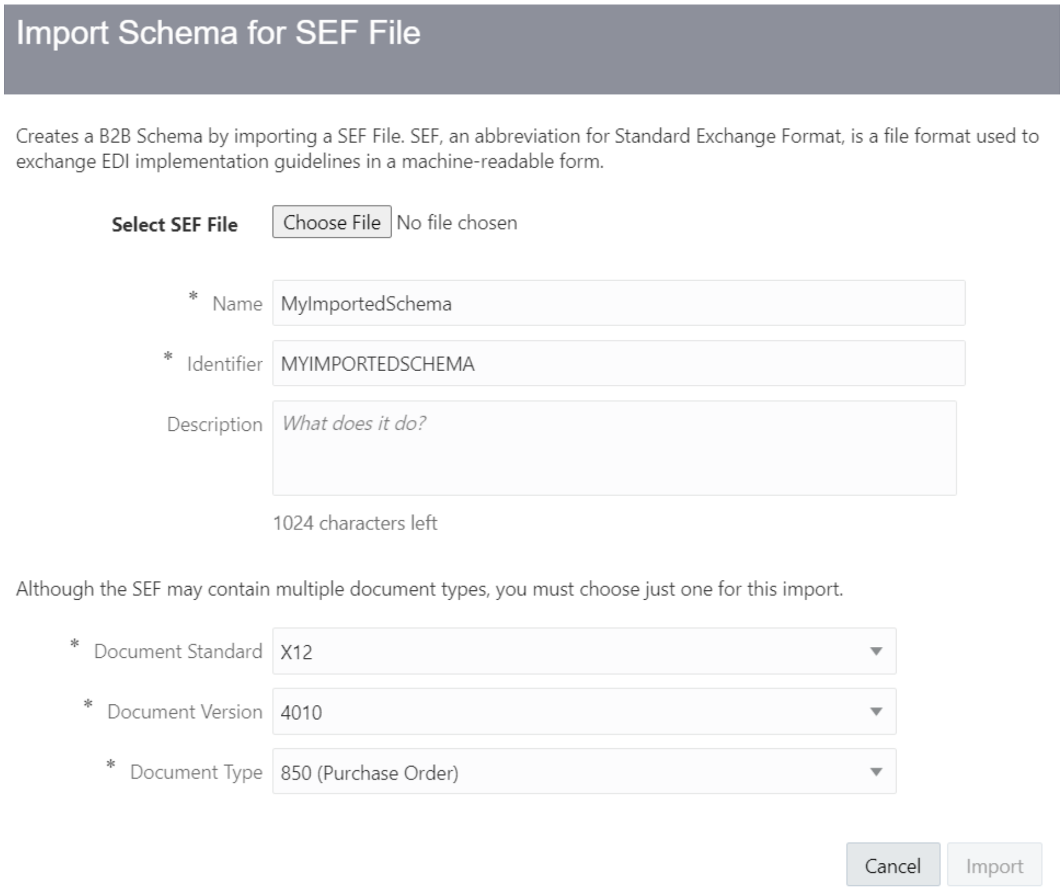


Create a Schema from an Imported SEF File

You can create a schema by importing a Standard Exchange Format (SEF) file. A SEF file exchanges EDI implementation guidelines in a machine-readable form.

1. On the B2B Schemas page, click **Import SEF**.
2. Enter the following details.

Element	Description
Select SEF File	Select the SEF file you want to import.
Name	Enter a name for the schema you want to create.
Identifier	This field is automatically populated with the document name. You can manually change this value.
Description	Enter an optional description of the customization details for this document.
Document Standard	Select the document standard. • EDIFACT • X12 EDI X12 is automatically selected for use.
Document Version	Select the document version.
Document Type	Select the document type.



Import Schema for SEF File

Creates a B2B Schema by importing a SEF File. SEF, an abbreviation for Standard Exchange Format, is a file format used to exchange EDI implementation guidelines in a machine-readable form.

Select SEF File	Choose File No file chosen
* Name	MyImportedSchema
* Identifier	MYIMPORTEDSCHEMA
Description	*What does it do?*
	1024 characters left

Although the SEF may contain multiple document types, you must choose just one for this import.

* Document Standard	X12
* Document Version	4010
* Document Type	850 (Purchase Order)

3. Click **Import**.

   An information message is displayed indicating that the SEF file has been imported successfully. The B2B schema is then created based on the SEF file.

Customize or Edit a B2B Schema

To create custom schemas, you can add new constructs (such as new segments and elements) to a standard B2B schema or edit existing constructs within it.

Topics:

- About B2B Schema Constructs
- Add New Segments and Other Schema Constructs
- Edit Properties of Segments and Other Schema Constructs
- Edit the Code List for an Element
- Edit Syntax Rules for Segments and Composites

About B2B Schema Constructs

A B2B schema represents the message format of an EDI X12 or EDIFACT document, which is a hierarchical structure based on four types of constructs:

- **Element:** The smallest unit that represents a single data field of a primitive type, such as, alphanumeric text, integer, decimal, date, time, or binary.
- **Composite:** A complex data type consisting of one or more elements.
- **Segment:** The next higher level construct, consisting of a sequence of elements and composites.
- **Loop or Loop Segment:** A container for a specific set of segments or child loops, which make its structure nested and hierarchical.

The schema editor in B2B for Oracle Integration displays each of these constructs in its own row. The rows for segments, composites, and loops are expandable, and display the elements they contain. The order of the rows in the schema defines the exact order in which each of the constructs must appear in an actual EDI X12 or EDIFACT document.

- An EDI X12 document, which is the topmost construct, is defined as an ordered set of segments and loops. The schema for all EDI X12 documents starts with a segment named `ST` and ends with the segment `SE` across all documents types. These segments are called transaction set envelope segments, and they cannot be deleted in the B2B schema editor.
- An EDIFACT document, which is the topmost construct, is also defined as an ordered set of segments and loops. The schema for all EDIFACT documents starts with a segment named `UNH` and ends with the segment `UNT` across all documents types. These segments are called transaction set envelope segments, and they cannot be deleted in the B2B schema editor.

Add New Segments and Other Schema Constructs

According to your organization's requirements, you can add new constructs to a standard B2B schema to customize it.

To add new constructs to a schema:

1. On the B2B Schemas page, hover over a schema and click **Edit** .
2. In the schema editor, hover over a row of required hierarchy and click **Actions** .
3. Select an option from the Actions list to add a new construct. The Details panel appears on the right, displaying the **Properties** tab, where you can enter the necessary information.

> **Note:**
>
> The options displayed in the Actions list vary according to the hierarchy of the rows. The Actions list for a segment or loop row has options to create new child elements, child composites, segments and loops, whereas the Actions list for a composite or element row has options to create only new composites and elements.

Edit Properties of Segments and Other Schema Constructs

B2B for Oracle Integration allows you to modify properties of existing constructs of a schema that's derived from a standard schema.

To edit the properties of constructs:

1. On the B2B Schemas page, hover over a schema and click **Edit** ✎.
2. In the schema editor, hover over a row you want to modify and click **Edit Details** ✎. The Details panel appears on the right, displaying the **Properties** tab. The property fields displayed in the tab vary for element, composite, segment, and loop rows. You can edit the field values to customize and deviate from the X12 standard according to your business requirements. The following sections describe the different property fields that are displayed for each construct type.

Element Properties

Property	Description
Name	A descriptive name for the element.
Element ID	The ID that identifies this element in the EDI X12 or EDIFACT element dictionary. This field is not editable.
Position	The element's position number within the parent segment or composite. This field is not editable.
Purpose	A detailed description of the element's purpose.
Requirement	• **Mandatory:** The element must have a nonempty value in an EDI document. • **Optional:** The element may have a value or be empty. • **Conditional:** The element may have a value or not depending on syntax rules defined on the parent segment or composite. This setting is preselected by the X12 or EDIFACT standard and typically not modified. To override this setting, modify the **Usage** field.
Usage	This field overrides the **Requirement** field. When this field is left blank (no value selected), it implies that the element is used and the selection made in the **Requirement** field is enforced. • **Must Use:** Overrides the selection in the **Requirement** field and treats the element as mandatory. • **Not Used:** Overrides the selection in the **Requirement** field and specifies that the element should remain empty (that is, not have a value). • **Recommended:** Follows the selection in the **Requirement** field, but suggests that the element is a preferred element. • **Not Recommended:** Follows the selection in the **Requirement** field, but suggests that the element is not a preferred element.
Type	The element's type as defined by the X12 standard in the EDI X12 element dictionary or the EDIFACT standard in the EDIFACT element dictionary. The type is one of the following: ID (Identifier), AN (Alphanumeric Text String), DT (date), TM (time), N(0-9) (Integer with implied decimal), R (Decimal), or B (Binary). This field is not editable.

Property	Description
Length (Min / Max)	The minimum and maximum characters (or numeric digits) that the element's value allows.
Repeat	Specifies if it's a repeating element. • `1`: Indicates that the element has a single value (non-repeating). • A number greater than one or the `>1` symbol: Indicates that this is a repeating element; that is, one element accommodates multiple values. The EDI X12 version 4012 and older versions do not support repeating elements; therefore, this setting is ignored for those versions.

Composite Properties

A composite has property settings similar to an element construct, except for the **Type** and **Length** fields.

Segment Properties

A segment has several property settings similar to an element construct. However, it doesn't have the **Element ID**, **Type**, and **Length** fields. In addition, there are the following differences.

Property	Description
Repeat	For a segment, this property indicates how many consecutive instances of the segment may occur in an EDI document. Segment repetition is supported in all X12 versions, unlike elements.
Table Area	Indicates the section of the EDI document to which this segment belongs. Typically, an EDI document is defined with a heading, summary, and detail sections, also called table areas.

Loop Properties

A loop has four property settings, namely **Requirement**, **Usage**, **Repeat**, and **Table Area**, which have similar meanings as those of a segment.

Edit the Code List for an Element

In addition to properties, you can also edit code lists for element constructs while customizing standard B2B schemas.

Some EDI X12 elements are restricted to a set of enumerated values. A code list displays the allowed values for an element. Typically, elements of the type ID (identifier) have a code list associated with them; however, any element can have a code list.

To edit an element's code list,

1. Hover over the element row and click **Edit Details**.
   The Details panel appears on the right, displaying the **Properties** tab.
2. Click **Code List**.
   If there is a code list defined, all code values are displayed in the table.

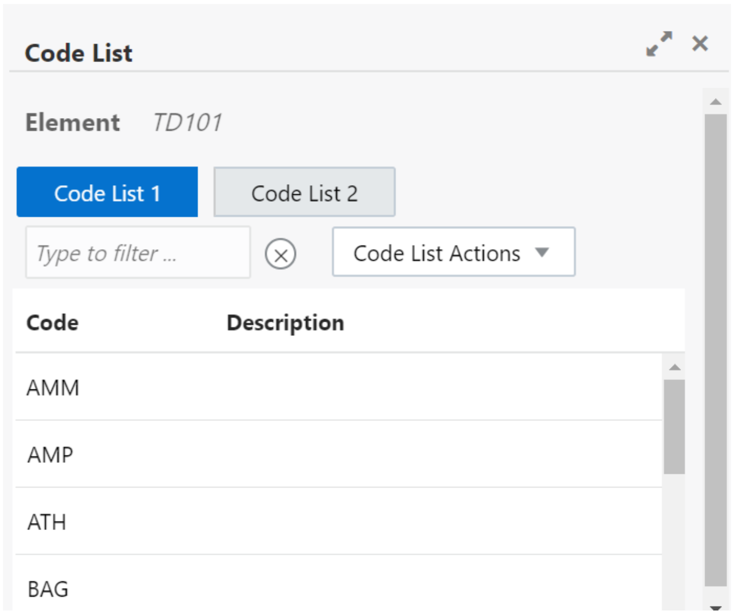
3. Hover over a row and use the buttons that appear to edit or delete the row or to add a new row.
While editing, you can add a description and a note to each code. While deleting, a row is first marked for deletion (which can be reversed with an **Undo**) and is actually deleted upon clicking **Save**.

> **Note:**
>
> While adding a new row, you can insert it at any position. The validation of an element's data against the code list happens regardless of the order. After a **Save** action, the code list is sorted alphabetically.

4. If you want to edit several rows in a code list, use the **Export CSV** action under **Code List Actions** to export the list to a comma-separated values file for editing in an external tool, such as a spreadsheet editor.
The CSV file has a header row identical to the code list table. After the editing is complete, you can import it back using the **Import CSV** action. This overwrites the existing code list completely with the codes defined in the CSV file.

Multiple Code Lists for an Element

Occasionally, you may find elements with multiple code lists. One such example is the element TD101 (Packaging Code) for the document type EDI X12 850 (any version). This element has two code lists defined as follows:



Code	Description
AMM	
AMP	
ATH	
BAG	

The element's data value is a concatenation of one code from each of the code lists. For example, a value of `AMM01` is a valid data value because it uses `AMM` from **Code List 1** and `01` from **Code List 2**.

Customizing a Code List

An element is considered customized (with respect to code lists) if any of its associated code lists have codes that are different from the standard X12 codes. However, only changing the description or adding notes is not considered a customization because it does not affect the element's allowed data values or the validation.

Edit Syntax Rules for Segments and Composites

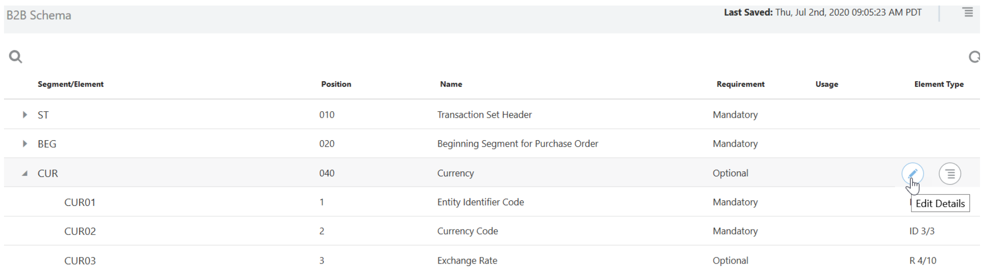
You can view and edit the syntax rules, if defined for a segment or a composite. These rules apply to data elements and are enforced during the validation of the input data.

> **Note:**
>
> You cannot edit a schema currently in use in an active document definition.

1. From the navigation pane on the Oracle home page, select **B2B** > **B2B Schemas**.

   The B2B schemas page shows all custom schemas you have created by cloning the standard schemas.

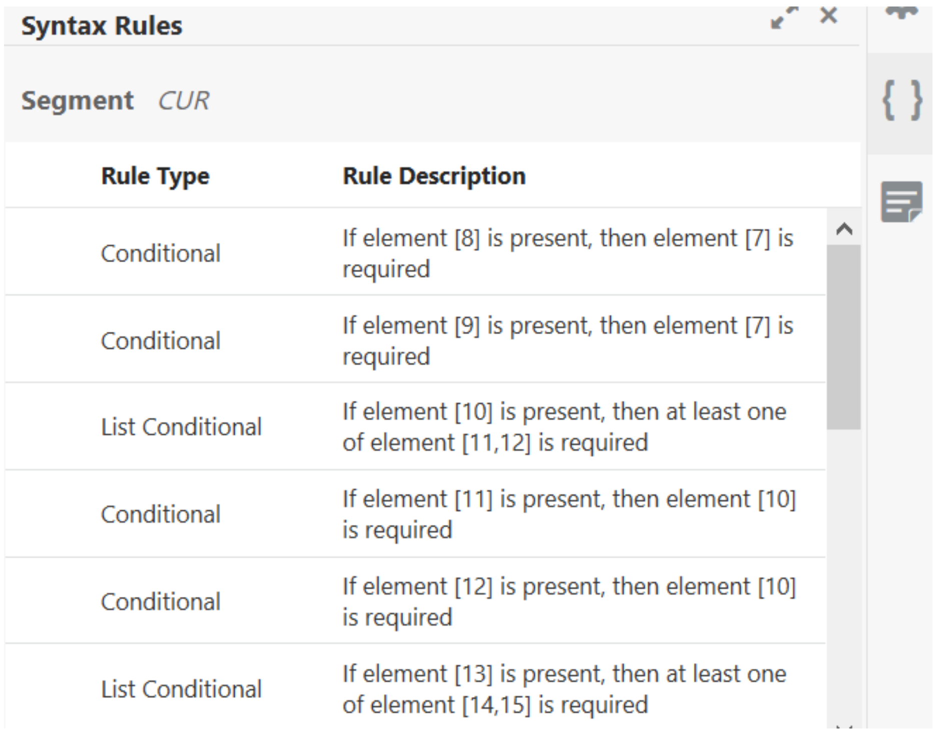
2. Open a custom schema, find the segment you want to view (for example, **CUR**), and select **Edit Details**.

   B2B Schema — **Last Saved:** Thu, Jul 2nd, 2020 09:05:23 AM PDT

Segment/Element	Position	Name	Requirement	Usage	Element Type
ST	010	Transaction Set Header	Mandatory		
BEG	020	Beginning Segment for Purchase Order	Mandatory		
CUR	040	Currency	Optional		
CUR01	1	Entity Identifier Code	Mandatory		
CUR02	2	Currency Code	Mandatory		ID 3/3
CUR03	3	Exchange Rate	Optional		R 4/10

3. Click { } in the Details pane to display the syntax rules enforced for the selected segment.

   Syntax Rules

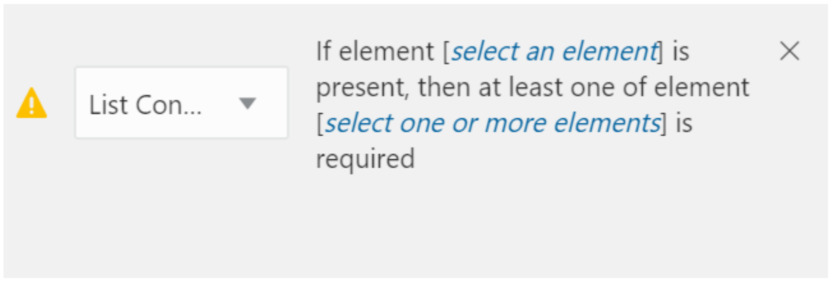
   Segment *CUR*

Rule Type	Rule Description
Conditional	If element [8] is present, then element [7] is required
Conditional	If element [9] is present, then element [7] is required
List Conditional	If element [10] is present, then at least one of element [11,12] is required
Conditional	If element [11] is present, then element [10] is required
Conditional	If element [12] is present, then element [10] is required
List Conditional	If element [13] is present, then at least one of element [14,15] is required

4. Hover over a syntax rule row to see options to edit, add, or delete a rule.
5. If you select the **Add** icon, a **Select** list is displayed with the following syntax rule types:

X12 Syntax Rule	Description
Pairing	If any of the elements are present, all are required.
Conditional	If one element is present, other elements are required.
List Conditional	If one element is present, at least one of the other elements is required.
Required	At least one of the elements is required.
Exclusion	Only one of the elements may be present.

   For each rule, you are presented with either one or two element selections into which you select elements.
6. For example, if you add the **List Conditional** rule, click **select an element** for the first element selection and click **select one or more elements** for the second element selection.

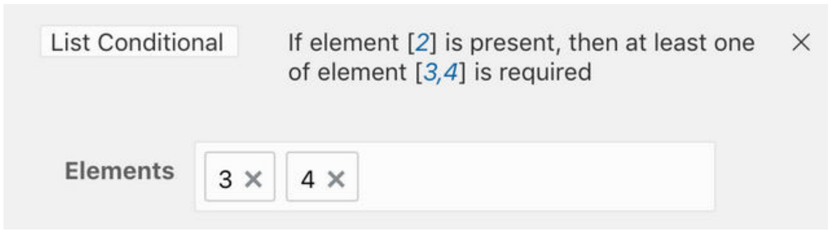


List Con...

If element [*select an element*] is present, then at least one of element [*select one or more elements*] is required

7. Select appropriate elements.

   The current selections are displayed in terms of the element position numbers.

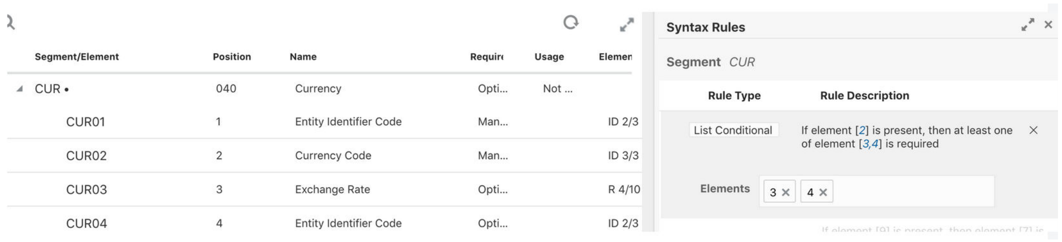


List Conditional

If element [*2*] is present, then at least one of element [*3,4*] is required

Elements 3 × 4 ×

   The element position numbers identify the corresponding child elements inside the selected segment or composite. For example, **2** means **CUR02** below and **3** and **4** mean **CUR03** and **CUR04**.



Segment/Element	Position	Name	Requir	Usage	Elemen
CUR •	040	Currency	Opti...	Not ...	
CUR01	1	Entity Identifier Code	Man...		ID 2/3
CUR02	2	Currency Code	Man...		ID 3/3
CUR03	3	Exchange Rate	Opti...		R 4/10
CUR04	4	Entity Identifier Code	Opti...		ID 2/3

Syntax Rules

Segment *CUR*

Rule Type	Rule Description
List Conditional	If element [*2*] is present, then at least one of element [*3,4*] is required

Elements 3 × 4 ×

8. Click the element numbers in the rule description to add or remove elements.

> **Note:**
>
> Deleting a syntax rule first marks it for deletion (which can be undone) and then deletes it when the B2B schema is saved.

Generate an Implementation Guide

You can generate implementation guides for standard or customized schemas to share with your trading partners.

Topics:

- About the Implementation Guide
- Generate the Implementation Guide for a Schema

About the Implementation Guide

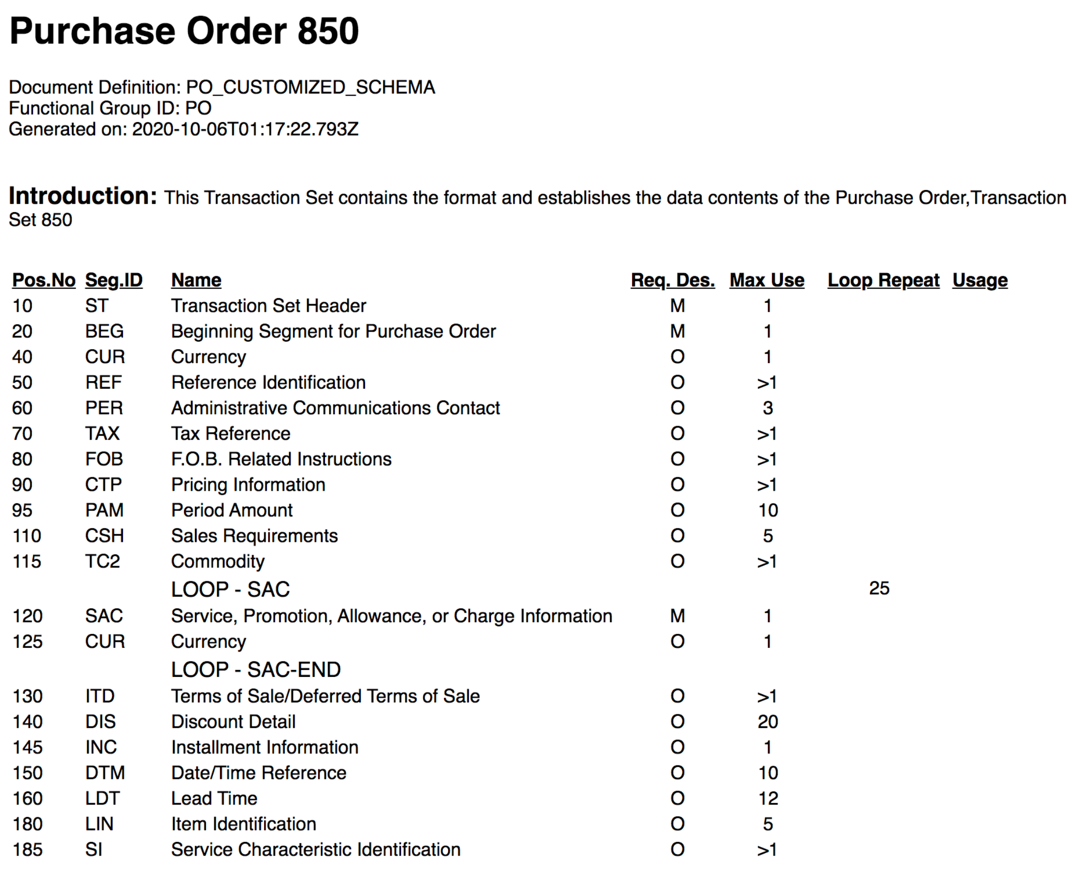
An implementation guide is a single, consolidated document with full details about a B2B schema.

Typically, a host company shares the implementation guide for an EDI schema with their trading partners. This guide is shared especially when a schema has been customized to deviate from a standard X12 document. It points out some of the customizations made to the standard schema (not all customizations are highlighted in the document).

In B2B for Oracle Integration, you can generate implementation guides for both standard and custom B2B schemas. The guide is generated as an HTML document, which you can edit to apply branding and print as a PDF file if necessary.

> **Note:**
>
> You cannot currently generate an implementation guide for EDIFACT documents.

The following image shows an example implementation guide:



Purchase Order 850

Document Definition: PO_CUSTOMIZED_SCHEMA
Functional Group ID: PO
Generated on: 2020-10-06T01:17:22.793Z

Introduction: This Transaction Set contains the format and establishes the data contents of the Purchase Order,Transaction Set 850

Pos.No	Seg.ID	Name	Req. Des.	Max Use	Loop Repeat	Usage
10	ST	Transaction Set Header	M	1		
20	BEG	Beginning Segment for Purchase Order	M	1		
40	CUR	Currency	O	1		
50	REF	Reference Identification	O	>1		
60	PER	Administrative Communications Contact	O	3		
70	TAX	Tax Reference	O	>1		
80	FOB	F.O.B. Related Instructions	O	>1		
90	CTP	Pricing Information	O	>1		
95	PAM	Period Amount	O	10		
110	CSH	Sales Requirements	O	5		
115	TC2	Commodity	O	>1		
		LOOP - SAC			25	
120	SAC	Service, Promotion, Allowance, or Charge Information	M	1		
125	CUR	Currency	O	1		
		LOOP - SAC-END				
130	ITD	Terms of Sale/Deferred Terms of Sale	O	>1		
140	DIS	Discount Detail	O	20		
145	INC	Installment Information	O	1		
150	DTM	Date/Time Reference	O	10		
160	LDT	Lead Time	O	12		
180	LIN	Item Identification	O	5		
185	SI	Service Characteristic Identification	O	>1		

Structure of an Implementation Guide

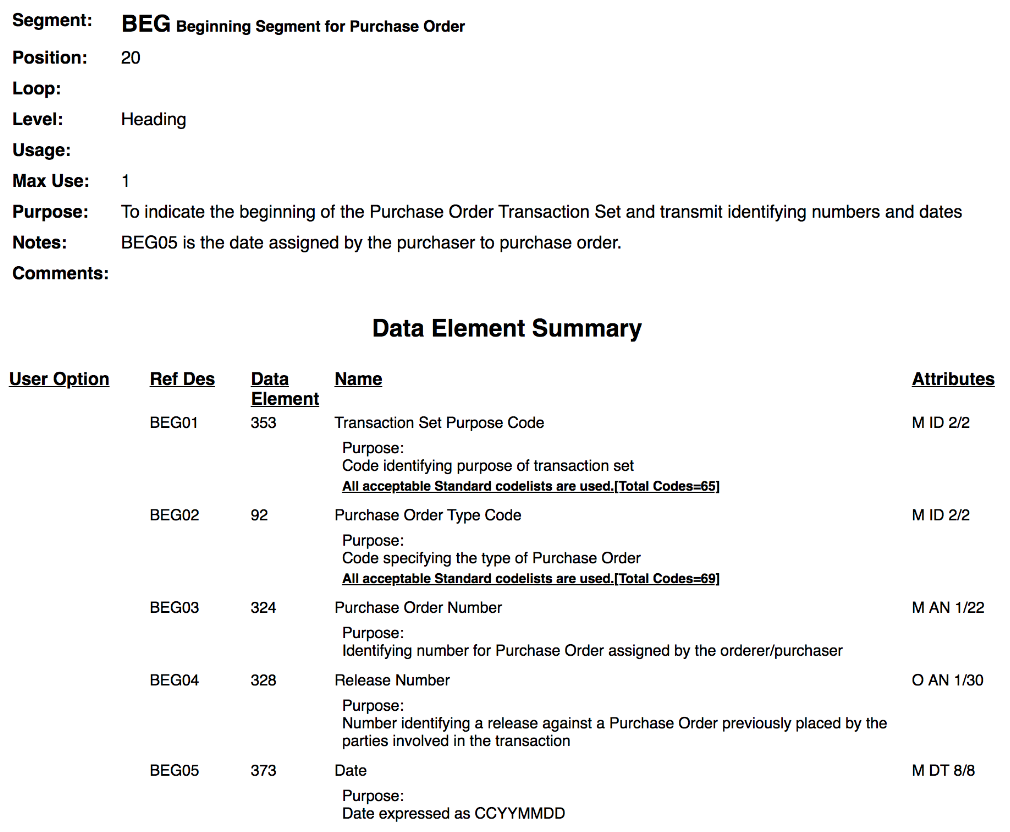
List of Segments and Loops

The guide begins with a listing of each segment and loop defined in the document schema. Each row defines a segment or loop with the following columns:

- Pos.No: The position number of the segment as defined in the X12 standard.
- Seg.ID: The segment identifier.
- Name: A descriptive name for the segment.
- Req.Des: The requirement designator (M = Mandatory, O = Optional, C = Conditional).
- Max Use: The Repeat property of the segment.
- Loop Repeat: The Repeat property of the loop, when the row represents a loop.
- Usage: The Usage property of the segment (Must Use, Not Used, Recommended, Not Recommended).

Details Section for Each Segment

Each segment has a details section that lists elements in the segment as shown in the following image:



Segment:	**BEG** **Beginning Segment for Purchase Order**
Position:	20
Loop:	
Level:	Heading
Usage:	
Max Use:	1
Purpose:	To indicate the beginning of the Purchase Order Transaction Set and transmit identifying numbers and dates
Notes:	BEG05 is the date assigned by the purchaser to purchase order.
Comments:	

Data Element Summary

User Option	Ref Des	Data Element	Name	Attributes
	BEG01	353	Transaction Set Purpose Code Purpose: Code identifying purpose of transaction set All acceptable Standard codelists are used.[Total Codes=65]	M ID 2/2
	BEG02	92	Purchase Order Type Code Purpose: Code specifying the type of Purchase Order All acceptable Standard codelists are used.[Total Codes=69]	M ID 2/2
	BEG03	324	Purchase Order Number Purpose: Identifying number for Purchase Order assigned by the orderer/purchaser	M AN 1/22
	BEG04	328	Release Number Purpose: Number identifying a release against a Purchase Order previously placed by the parties involved in the transaction	O AN 1/30
	BEG05	373	Date Purpose: Date expressed as CCYYMMDD	M DT 8/8

The Data Element Summary table lists the elements declared in the segment and has the following columns:

- User Option: The Usage property of the element (Must Use, Not Used, Recommended, Not Recommended).
- Ref Des: The reference designator of the element, which is a convention to represent an element. For example, BEG01 is the first element in the segment BEG. If an element is a composite element, the child elements in the composite use the convention REF04-01, where 01 is the sub-element position, 04 is the composite element's position, and REF is the segment.
- Data Element: The identifier of the element's definition.
- Name: A descriptive name assigned to the element.
- Attributes: Includes more properties of the element's definition. A value like `M ID 2/2` is made of the following parts:
  - The requirement designator for the element (M = Mandatory, O = Optional, C = Conditional).
  - Element's data type (ID=Identifier, AN=Alphanumeric String, N0=integer number, etc.).
  - Element's minimum and maximum character length. A value `1/30` implies that the element's value must be between 1 and 30 characters to be valid.

Generate the Implementation Guide for a Schema

View or download the implementation guide for a schema in the HTML format.

To generate an implementation guide:

1. **For a standard schema:** On the B2B Schemas page, click **Generate**, then click **Implementation Guide**.
2. **For a custom schema:** On the B2B Schemas page, hover over a schema row and click **Actions** ☰, then select **Generate Implementation Guide**.
3. On the Generate Implementation Guide page, click **Download** or **View** to generate the document in the HTML format.

5 B2B Tracking

You can track the B2B message interaction between trading partners on the Track B2B Messages page.

Topics:

- Track B2B Messages

Track B2B Messages

You can track the B2B message interaction between trading partners on the Track B2B Messages page. This page provides B2B-specific message tracking information that is not available on the Track Instances page for integrations.

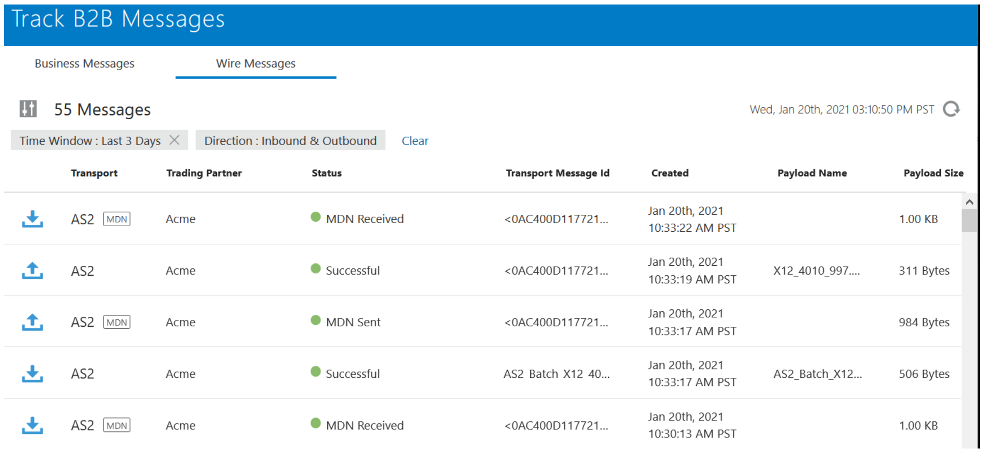
> **Note:**
>
> This feature is only available on Oracle Integration Generation 2.

1. In the left navigation pane, click **Home** > **Monitoring** > **B2B Tracking**.
   The **Wire Messages** tab of the Track B2B Messages page is displayed.

   Wire messages are raw messages exchanged between trading partners. Wire messages flow back and forth between trading partners across the dotted line shown in the following document transaction example. For the following example, an interaction is initiated by an inbound message sent from a trading partner to the host trading partner (site in which Oracle Integration is installed). There are four message interactions in this example. All four are tracked on the Track B2B Messages page.

   a. The host trading partner receives an invoice document (EDI 810) from a trading partner.
   b. The host trading partner performs the following actions:
      i. Sends a message disposition notification (MDN) to the trading partner to acknowledge receipt of the invoice.
      ii. Parses the EDI document to a format understood by Oracle Integration.
   c. The host trading partner sends a functional acknowledgment (EDI 997) to the trading partner.
   d. The trading partner sends a message to the host trading partner acknowledging receipt of the EDI 997.

   These B2B message interaction details are captured on the Track B2B Messages page.



55 Messages

Wed, Jan 20th, 2021 03:10:50 PM PST

Time Window : Last 3 Days
Direction : Inbound & Outbound
Clear

	Transport	Trading Partner	Status	Transport Message Id	Created	Payload Name	Payload Size
⬇	AS2 MDN	Acme	● MDN Received	<0AC400D117721...	Jan 20th, 2021 10:33:22 AM PST		1.00 KB
⬆	AS2	Acme	● Successful	<0AC400D117721...	Jan 20th, 2021 10:33:19 AM PST	X12_4010_997....	311 Bytes
⬆	AS2 MDN	Acme	● MDN Sent	<0AC400D117721...	Jan 20th, 2021 10:33:17 AM PST		984 Bytes
⬇	AS2	Acme	● Successful	AS2 Batch X12 40...	Jan 20th, 2021 10:33:17 AM PST	AS2_Batch_X12...	506 Bytes
⬇	AS2 MDN	Acme	● MDN Received	<0AC400D117721...	Jan 20th, 2021 10:30:13 AM PST		1.00 KB

The left column identifies the message direction. Inbound messages (those received by the host Oracle Integration trading partner) are identified by

and outbound messages (those sent from the host Oracle Integration trading partner) are identified by

.

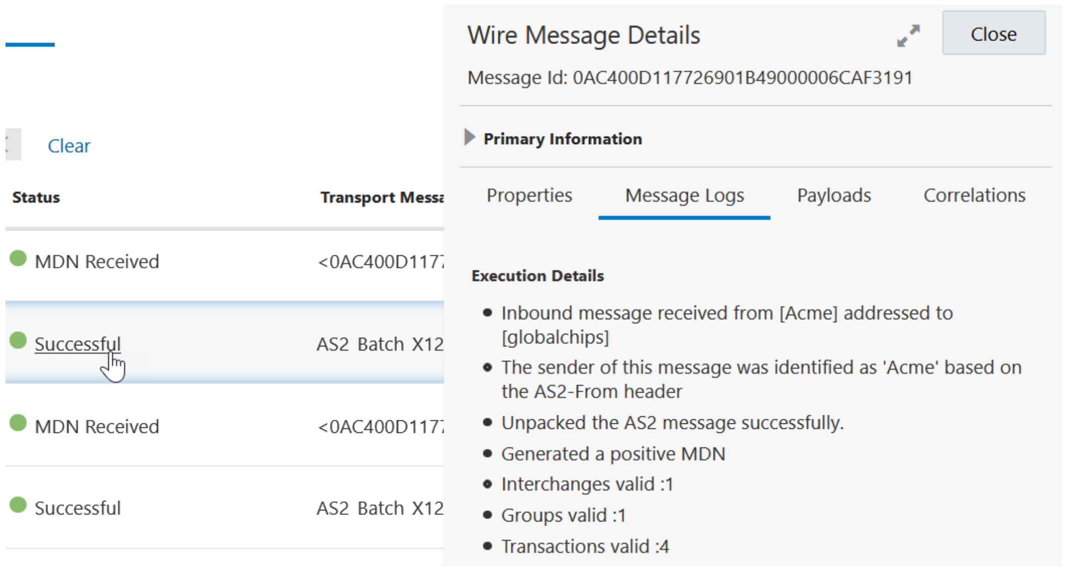
The **Payload Name** column shows the payload file that was transmitted.

2. Click

   to filter B2B message display by the last message created, time window, trading partner name, direction, message type, status, transport protocol, message ID, transport message ID, integration instance ID, error code, payload name, and parent message ID.

3. Click an entry in the **Status** column or click

   . For this example, a B2B document sent in the inbound direction to the host trading partner is selected. Note also that MDN acknowledgment messages that were sent or received are also available for selection.
   This opens the Wire Message Details page. The **Message Logs** tab is displayed by default.



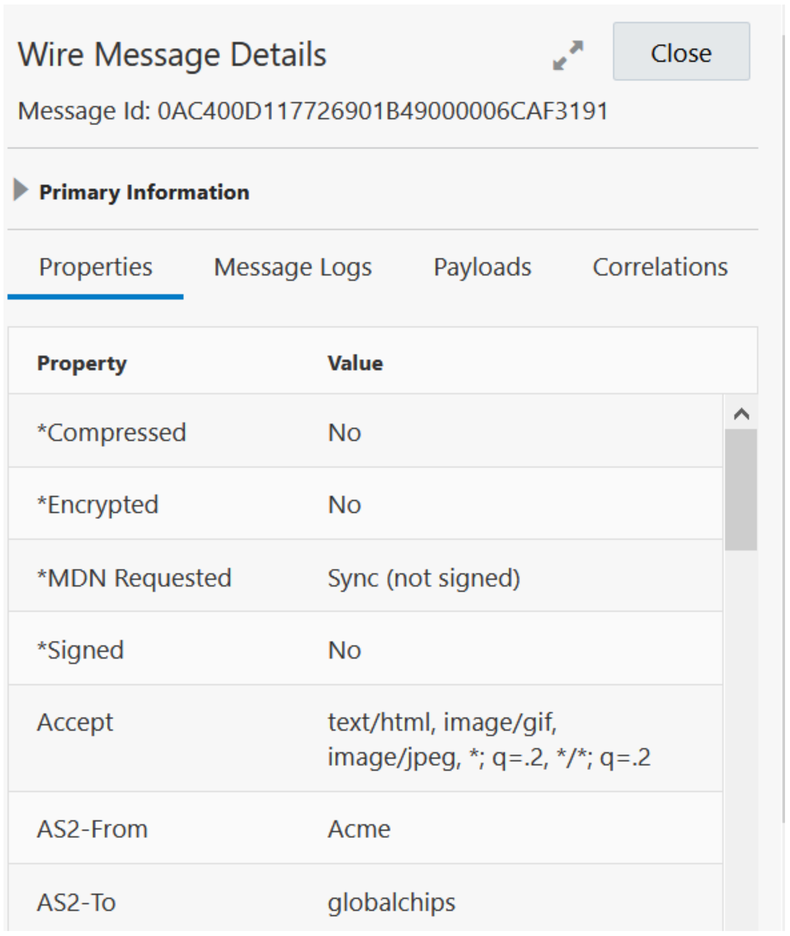
Wire Message Details

Message Id: 0AC400D117726901B49000006CAF3191

Execution Details

- Inbound message received from [Acme] addressed to [globalchips]
- The sender of this message was identified as 'Acme' based on the AS2-From header
- Unpacked the AS2 message successfully.
- Generated a positive MDN
- Interchanges valid :1
- Groups valid :1
- Transactions valid :4

4. View the following details in the four available tabs:
   - **Message Logs** (displayed by default): The message log is similar to the activity stream available for integrations. It provides details such as the following:
     – The message sent from the trading partner to the host trading partner (since this example is an inbound message). The trading partner names shown are based on the headers that you mapped to the B2B identifiers defined in the agreement. See Create Trading Partners.
     – The identity of the trading partner that sent the message.
     – The unpacking of the message (preprocessing tasks such as message decryption, signature verification, and others).
     – The status of the MDN acknowledgment message sent to the trading partner (for this example, the status is successful).
     – Additional postprocessing tasks.
   - **Properties**: Shows header properties and values (for example, HTTP and MIME).



Wire Message Details

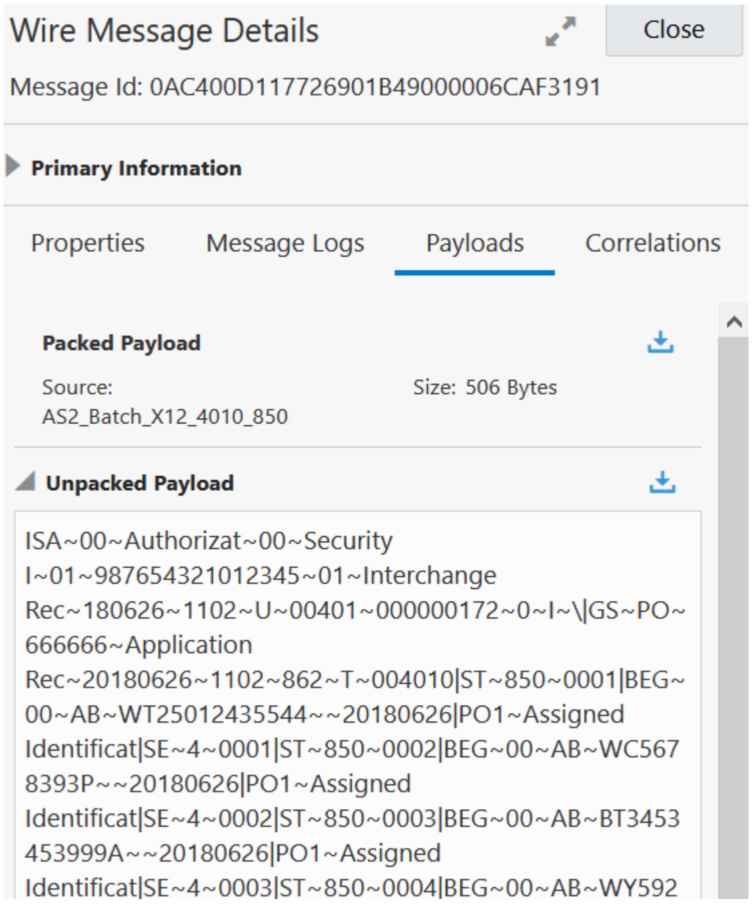
Message Id: 0AC400D117726901B49000006CAF3191

Property	Value
*Compressed	No
*Encrypted	No
*MDN Requested	Sync (not signed)
*Signed	No
Accept	text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
AS2-From	Acme
AS2-To	globalchips

- **Payloads**: Shows the following details:
  - **Packed Payload**: Shows the physical EDI payload sent from the trading partner for this example. The payload may be encrypted and signed if both were configured by the trading partner. A signed payload ensures that your message cannot be repudiated.
  - **Unpacked Payload**: Shows the raw, unencrypted version of the payload sent from the trading partner. The payload format may be signed if it was configured by the trading partner.

  Click

  to download the packed or unpacked version of the payload.


Wire Message Details

Close

Message Id: 0AC400D117726901B49000006CAF3191

Primary Information

Properties | Message Logs | Payloads | Correlations

Packed Payload

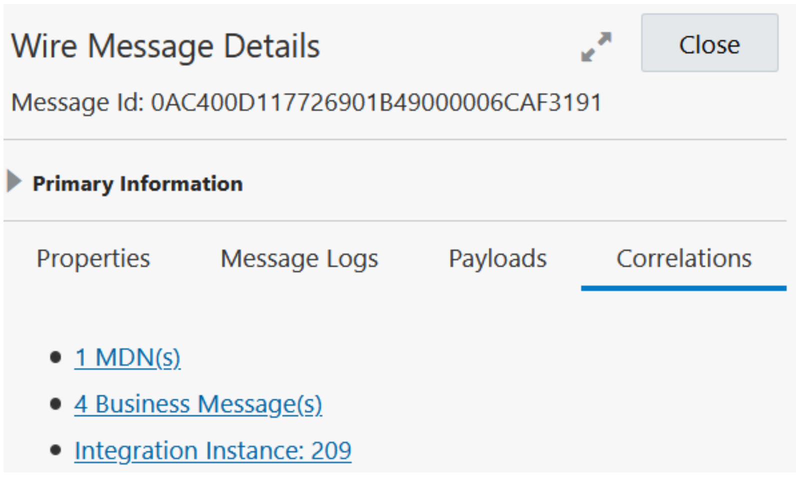
Source: AS2_Batch_X12_4010_850

Size: 506 Bytes

Unpacked Payload

```
ISA~00~Authorizat~00~Security
I~01~987654321012345~01~Interchange
Rec~180626~1102~U~00401~000000172~0~I~\|GS~PO~
666666~Application
Rec~20180626~1102~862~T~004010|ST~850~0001|BEG~
00~AB~WT25012435544~~20180626|PO1~Assigned
Identificat|SE~4~0001|ST~850~0002|BEG~00~AB~WC567
8393P~~20180626|PO1~Assigned
Identificat|SE~4~0002|ST~850~0003|BEG~00~AB~BT3453
453999A~~20180626|PO1~Assigned
Identificat|SE~4~0003|ST~850~0004|BEG~00~AB~WY592
```

- **Correlations**: Shows the messaging interaction between trading partners. This tab provides insight into the entire message interaction between trading partners.


Wire Message Details

Close

Message Id: 0AC400D117726901B49000006CAF3191

Primary Information

Properties | Message Logs | Payloads | Correlations

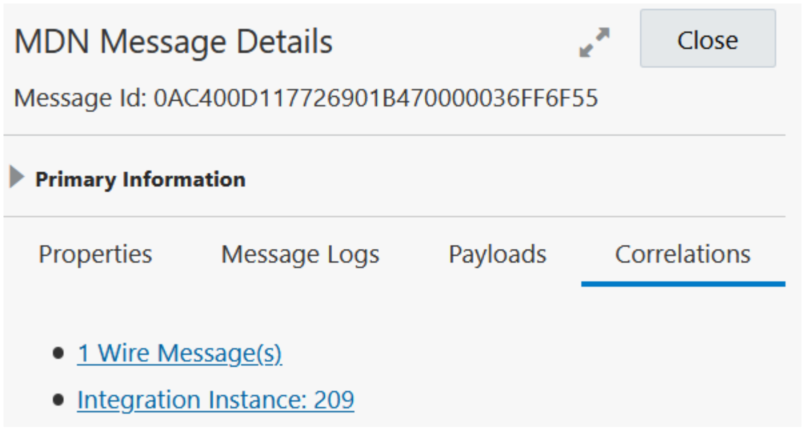
- 1 MDN(s)
- 4 Business Message(s)
- Integration Instance: 209

  – **MDN(s)**:

    a. Click **MDN(s)** to show the B2B Messages **tab** filtered to show only details about the MDN message that the host trading partner sent back to the trading partner to acknowledge receipt of the initial inbound message (for this example).

b. Click the status in the **Status** column (for example, **MDN Sent**) to show property, message log, payload, and correlation links with details about the sent MDN message.

c. Click **Correlations** to show the message correlation in this transaction from the perspective of the MDN message.



Message Id: 0AC400D117726901B470000036FF6F55

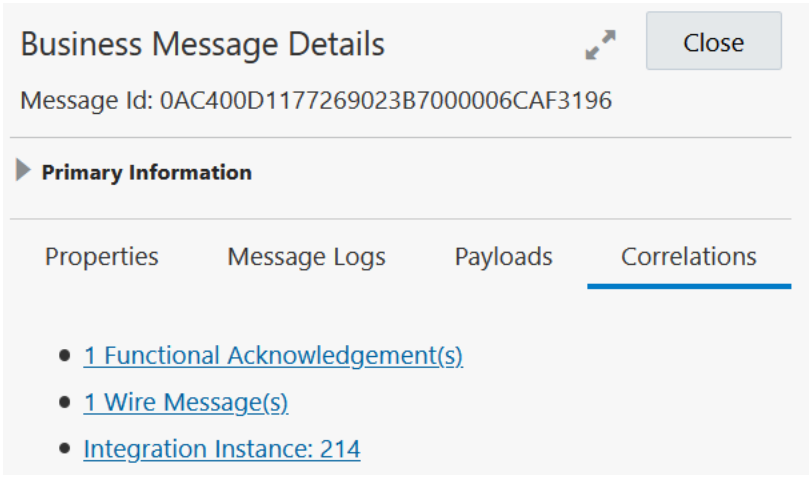
- 1 Wire Message(s)
- Integration Instance: 209

– **Business Message(s)**:

a. Click **Business Message(s)** to show the **Business Messages** tab of the Track B2B Messages page filtered to show only details about the business message in this interaction. Depending upon the message interaction, multiple business messages can be displayed. When the payload is translated with the EDI parser, an XML message is created that corresponds to this business message.
If the business message includes an invoice, a **Customer Identifier** column is displayed that provides the invoice number extracted from the business message.

b. Click the status in the **Status** column to show property, message log, payload, and correlation details about the business message.

c. Click **Correlations** to show the message correlation in this transaction from the perspective of the business message.



Message Id: 0AC400D1177269023B7000006CAF3196

- 1 Functional Acknowledgement(s)
- 1 Wire Message(s)
- Integration Instance: 214

- **Integration Instance: *number***:
  a. Click **Integration Instance:** to access the backend integration that includes this B2B message on the Track Instances page.

6

B2B Action XML Schema Reference

This section provides reference information about the B2B action XML schema.

Topics:

- B2B Action Input and Output Schema Reference

B2B Action Input and Output Schema Reference

While configuring data mappings for the B2B action, you use the `TranslateInput` and `TranslateOutput` elements and map the incoming and outgoing data to or from these elements.

This section describes the schema elements contained within the `TranslateInput` and `TranslateOutput` data elements. These schema elements drive the behavior of the B2B action.

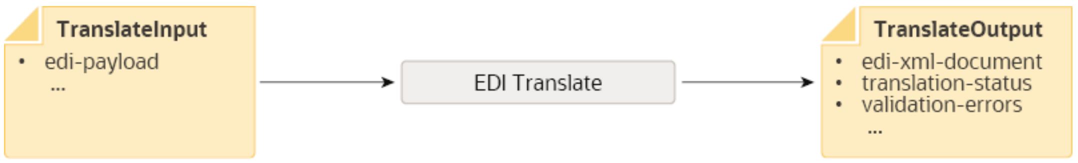
Topics:

- Schema Elements for Inbound EDI
- Sample TranslateOutput XML for Inbound EDI
- Schema Elements for Outbound EDI
- Sample TranslateOutput XML for Outbound EDI

Schema Elements for Inbound EDI

In the inbound EDI use case, the input to the B2B action has the EDI payload element and the output, which is generated after parsing, has the XML form of the EDI document.

The `TranslateInput` data element represents the input message to the B2B action, and the `TranslateOutput` represents the output of the action. Therefore, at a minimum, `TranslateInput` needs to contain the `edi-payload` element to which you'll assign a value using the input map action. The `TranslateOutput` data element then produces the `edi-xml-document` element, returning the parsed data in the XML format along with validation errors, if any.



The following tables describe each element contained in `TranslateInput` and `TranslateOutput` for the inbound EDI scenario.

Elements in TranslateInput

Element	Description
`edi-payload`	You should assign a base64-encoded string value generated from a native EDI X12 payload (starting with the ISA segment and ending with the IEA segment) to this element. Base64 encoding is required because EDI may contain unprintable characters used as delimiters or even binary content such as images. In case the EDI X12 payload contains a batch of transactions, you must use a stage file action along with the B2B action to process the batch.
`tracking-info`	You'll require this element only when you use the B2B action with the stage file action for batch processing. This element contains the tracking information for each EDI transaction that is passed from the stage file action to the B2B action.
`edi-encoding`	The character encoding of the EDI X12 payload (default is UTF-8).

`accept-message-on-errors`
and
`reject-message-on-errors`
	By default, any syntactical error in the input EDI payload is treated as a validation error in translation. You can alter this behavior in two ways:
	1. By turning off validation checks for the current message. Or:
	2. By keeping the validation turned on for the general case. But, be lenient for certain types of validation errors, allowing the message to be processed successfully even if those validation errors were to occur.
	These elements offer the second option listed above by providing a fine-grained control over the types of validation errors to ignore. You may need to have this fine-grained control in specific situations where you want to forward the message to a back-end application; for example, despite the B2B layer detecting invalid data.
	Both these elements are of a string type, and the format of the string allows you to specify one or more validation error types. These elements are semantic opposites of one another.
	The format is as follows:
	`error#<error_code1>,error#<error_code2>,...`
	It is a comma-separated string, where each token starts with `error#` followed by an error code such as `B2B-01752` (for example, a token is: `error#B2B-01752`).
	The `accept-message-on-errors` element defines validation errors to ignore, while the `reject-message-on-errors` element defines validation errors to treat as real errors.
	When a validation error is treated as a real error, the B2B action also generates an acknowledgment message (EDI X12 997) with a rejected flag. However, when a validation error is ignored, the 997 is flagged with the `Accepted, but errors were noted` status.

Element	Description
`validate`	By default, any syntactical error in the input EDI payload is treated as a validation error in translation. You can alter this behavior in two ways: 1. By turning off validation checks for the current message. Or: 2. By keeping the validation turned on for the general case. However, be lenient for certain types of validation errors, allowing the message to be processed successfully even if those validation errors were to occur. This element offers the first option listed above. Setting this element to `false` turns off validation checks, which effectively has the same effect as deselecting the check box **Perform validations on input data?** in the Configure B2B Action Wizard. However, this element overrides the check box setting and can be set for every message (whereas the check box setting is used for all messages). If this element is not set, the check box setting in the wizard controls whether validations are turned on or off.
`functional-ack-required`	If you set this element to `false`, the B2B action does not generate functional acknowledgment messages. If set to `true` or not set, a functional acknowledgment message is included in the `TranslateOutput` element when the B2B action processes the last message in a batch (that is, the last transaction set inside a functional group).
`passthrough-errors`	You'll require this element only when you use the B2B action with the stage file action for batch processing. This element simply transfers certain errors from the stage file action to the B2B action so that a proper functional acknowledgment message can be generated in case of errors. You must not set this element when using the B2B action in standalone mode.
`routing-info`	You'll require this element only when you use the B2B action with the stage file action for batch processing. You must not set this element when using the B2B action in standalone mode.
`input-source-context`	To use this optional element, you can assign any identifier string that the B2B action simply copies to the `TranslateOutput` element. For example, if the B2B action is processing a message from a file, set the file name to this element. The purpose of the element is to make the value available in the output. Therefore, it can be relayed to further actions in your integration or stored inside a tracking variable.

Elements in TranslateOutput

Element	Description
`edi-xml-document`	This element is returned with the translated message in the XML form. See Sample TranslateOutput XML for Inbound EDI.

Element	Description

`edi-xml-document` > `headers` > `interchange-ctrl`
`edi-xml-document` > `headers` > `group`
	These header elements are returned with values parsed from the EDI X12 ISA and GS envelopes.

`edi-xml-document` > `func-ack-report`	This element is returned only when the document type is a 997 (functional acknowledgment), and carries special content specific to a 997. Here are the child elements: • `original-tracking-info`: The tracking identifier of the original outbound message reconstructed from this functional acknowledgment. This tracking identifier has a substring match with the `tracking-info` that is generated in `TranslateOutput > tracking-info` for the outbound EDI case. This is used to correlate an incoming 997 to an outbound message sent to a trading partner originally. For example, the value for `original-tracking-info` is `tc=1013;gc=1013;sn=SenderID;rc=ReceiverID;st=A;` and the corresponding value for `TranslateOutput > tracking-info` is `tc=1013;gc=1013;sn=SenderID;rc=ReceiverID;ic=000000015` (these values have a substring match, that means, the part of the string `tc=1013;gc=1013;sn=SenderID;rc=ReceiverID` from both values have an exact match). • `func-ack-status`: The overall acceptance status as reported by the trading partner. Values are: `Success`, `Error`, or `Warning`, which correspond to `Accepted`, `Rejected`, or `Accepted But Errors Were Noted` status. • `func-ack-details`: The interpretation of the functional acknowledgment in a human-readable, plain text format.

`edi-xml-document` > `transaction-data`	This element is returned with the hierarchical structure of data segments and elements inside the EDI message, starting with the `ST` segment and ending with the `SE` segment.
	Each segment or loop inside the EDI X12 data becomes a parent XML element and its constituent EDI elements become child XML elements.
	For example, this may appear inside the `transaction-data` element:
	`<ST>`
	    `<ST01>850</ST01>`
	    `<ST02>1234</ST02>`
	`</ST>`
	The above XML fragment represents the data values parsed from the EDI X12 data that has this line: `ST*850*1234~`
	The XML elements `<ST01>` and `<ST02>` are assigned values based on their corresponding element positions occurring inside the EDI data.

`edi-xml-document` > `trailers` > `group`
`edi-xml-document` > `trailers` > `interchange-ctrl`
	The trailer elements are returned with values parsed from the EDI X12 IEA and GE envelopes.

Element	Description
`functional-ack-present`	This element is returned as `true` if an EDI X12 997 functional acknowledgment has been generated in the element `functional-ack`. This element is returned as `false` if no acknowledgment message was needed or generated (if `TranslateInput > functional-ack-required` was set to `false`, then an acknowledgment message is never generated). The B2B action automatically generates a functional acknowledgment message as follows: • For a singleton transaction (for example, when there is a single purchase order inside an EDI message or file): A functional acknowledgment is generated immediately, at the same time as the transaction is processed. • For a batch transaction (for example, when there are fifty purchase orders inside an EDI message or file): A functional acknowledgment is generated when the last transaction inside a batch is processed. A batch means one EDI X12 functional group containing multiple transactions (that is, multiple pairs of `ST`/`SE` segments). The functional acknowledgment then reports the status of all the transactions inside that batch, by including multiple AK2 and AK5 segments. According to the EDI X12 standard, only one functional acknowledgment is expected per batch.
`functional-ack`	If `functional-ack-present` is returned as `true`, the `functional-ack` element is returned with a Base64-encoded EDI X12 997 (functional acknowledgment) document that is ready to be sent to the trading partner. You must apply a Decode64 function to get the native EDI content at the point where you are about to send it out (for example, before writing it to a file to deliver to the trading partner).
`functional-ack-tracking-info`	This element is returned with a string representing a unique transmission identifier for the generated functional acknowledgment message. The string includes control numbers used for the functional acknowledgment. The value is useful for tracking purposes in case the trading partner is unable to process the 997 document.
`validation-errors-present`	This element is returned as `true` if validation errors were detected while parsing the input payload.

`validation-errors` > `error`
`validation-errors` > `validation-error-report`
	The `error` element is returned with validation errors, with each validation error in an XML structure; the error code and error message are returned as separate XML elements. A maximum of 100 validation errors are reported.
	The `validation-error-report` element is returned with a concatenated list of up to 100 validation errors, separated by line break characters.

`tracking-info`	This element is returned with a string representing a unique transmission identifier occurring in the input EDI. This string includes control numbers used in the input EDI message (interchange/group/transaction-set control numbers). This value is useful for tracking purposes.

Element	Description
`translation-status`	This element is returned with the values `Success`, `Warning`, or `Error` as follows: `Success`: Indicates that the input EDI was parsed successfully and no validation errors were detected (also occurs when validations are turned off). `Warning`: Indicates there were validation errors but those were minor or that the user chose to treat certain errors leniently, controlled by the `TranslateInput > accept-message-on-errors` value. `Error`: Indicates critical validation errors were detected while translating and the message is marked as rejected in the functional acknowledgment contents. The integration should not send the message to a backend system for further processing; instead, it should route the message to error handing.
`transaction-summary`	This element is returned with the overall counts of transactions parsed with success, warning, or errors across a batch of transactions.
`input-source-context`	This element is returned as a verbatim copy of the `TranslateInput > input-source-context`, so that it is visible in the integration actions that follow.

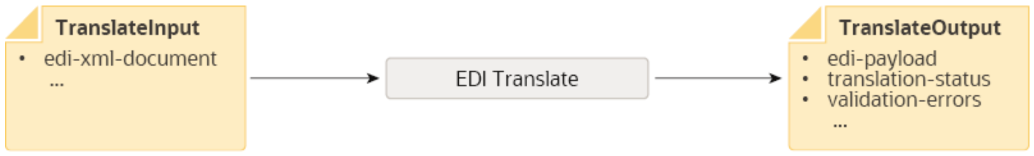
Sample TranslateOutput XML for Inbound EDI

This section provides a sample TranslateOutput XML file for inbound EDI.

```
<ns0:executeResponse xmlns:ns0="http://xmlns.oracle.com/cloud/adapter/
ediAdapter/EDI_Translate_REQUEST/types">
    <TranslateOutput xmlns="http://xmlns.oracle.com/b2b/X12/4010/
transactionset-850/default" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
        <edi-xml-document format="X12-4010-850" ver="1.0.20200326">
            <headers>
                <interchange-ctrl ack-requested-code="0" auth-info=""
auth-info-qualifier="00" ctrl-standards-id="U" ctrl-version-num-
code="00401" date="190312" element-separator="|" interchange-ctrl-
num="000001894" receiver-id="22222" receiver-id-qualifier="01" sec-
info="" sec-info-qualifier="00" segment-terminator="0xa" sender-
id="111111T" sender-id-qualifier="01" subelement-separator="~"
time="0845" usage-indicator-code="T"/>
                <group app-receivers-code="ACME" app-senders-
code="0124578" date="20190312" func-identifier-code="PO" group-ctrl-
num="0123456" resp-agency-code="X" time="084515" version-identifier-
code="004010"/>
            </headers>
            <transaction-data>
                <ST>
                    <ST01>850</ST01>
                    <ST02>1013</ST02>
                </ST>
                <BEG>
                    <BEG01>99</BEG01>
                    <BEG02>BA</BEG02>
                    <BEG03>Purchase Order Number</BEG03>
                    <BEG05>20061103</BEG05>
                    <BEG06>Contract Number</BEG06>
                    <BEG09>01</BEG09>
                </BEG>
                ....
                <SE>
                    <SE01>25</SE01>
                    <SE02>1013</SE02>
                </SE>
            </transaction-data>
            <trailers>
                <group count-of-txnsets="1" group-ctrl-num="1013"/>
                <interchange-ctrl count-of-groups="1" interchange-ctrl-
num="100000020"/>
            </trailers>
        </edi-xml-document>
        <functional-ack-present>true</functional-ack-present>
        <functional-ack>SVNBfDAwfCAgICAgICAgICB8MDB8ICAgICAgICAgI...</
functional-ack>
        <functional-ack-tracking-
info>tc=1016;gc=1016;sn=22222;rc=111111T;ic=000000016;</functional-ack-
tracking-info>
        <validation-errors-present>true</validation-errors-present>
        <validation-errors>
            <error>
                <code>B2B-01757</code>
                <category>data_codelist</category>
                <summary>Element BEG01 (element id 353) contains an invalid
identification code [99] that is not listed in the allowed codes for this
element (for example: 00, 44, 01, 45, 02, 46 ...).</summary>
                <location>segment-BEG &gt; BEG01 (element id 353) | segment
position 2 (starting with ST segment) | element position 1 | character
position 174</location>
            </error>
            <validation-error-report>[1] Error code: B2B-01757 | category:
data_codelist | message: (Severe) Element BEG01 (element id 353) contains an
invalid identification code [99] that is not listed in the allowed codes for
this element (for example: 00, 44, 01, 45, 02, 46 ...). | segment-BEG &gt;
BEG01 (element id 353) | segment position 2 (starting with ST segment) |
element position 1 | character position 174</validation-error-report>
        </validation-errors>
        <tracking-
info>tc=1234;gc=0123456;sn=111111T;rc=22222;ic=000001894;seq=1;tot=1;</
tracking-info>
        <translation-status>Error</translation-status>
    </TranslateOutput>
</ns0:executeResponse>
```

Schema Elements for Outbound EDI

In the outbound EDI use case, the input to the B2B action has the XML form of the EDI document and the output has the document assembled into the EDI format.



The following tables describe each element contained in `TranslateInput` and `TranslateOutput` for the Outbound EDI scenario.

Elements in TranslateInput

Element	Description
`edi-xml-document`	The XML form of the data to use to form the output EDI message during translation. At the minimum, you must assign values to the following child XML elements in the input: • `headers > interchange-ctrl`: – The attribute `sender-id-qualifier` must contain a value for the interchange sender ID qualifier. This value is inserted into the ISA05 element. – The attribute `sender-id` must contain a value for the interchange sender ID qualifier. This value is inserted into the ISA06 element. – The attribute `receiver-id-qualifier` must contain a value for the interchange receiver ID qualifier. This value is inserted into the ISA07 element. – The attribute `receiver-id` must contain a value for the interchange receiver ID qualifier. This value is inserted into the ISA08 element. • `headers > group`: – The attribute `app-senders-code` must contain a value for the application sender's code. This value is inserted into the GS02 element. – The attribute `app-receivers-code` must contain a value for the application receiver's code. This value is inserted into the GS03 element. • `transaction-data:` – The child elements inside it represent the structure of the business message, such as a purchase order. Mapping these elements require knowledge of the EDI X12 segments and elements that form the business message. You must assign values to mandatory segments and elements, and assign values to conditional elements based on the presence of other elements defined by the syntax rules specified for each segment. Other elements and attributes inside headers and trailers are assigned default values, so these are optional. **Note:** The B2B action automatically generates unique control numbers for the interchange, group, and transaction set. **Note:** **Customizing Delimiters for Outbound EDI** By default, the `element-separator` is `*`, the `segment-delimiter` is `~`, and the

Element	Description
	`subelement-separator` is `:`. You can override them by specifying values in the input XML as follows: • `headers` > `interchange-ctrl` > `attribute 'element-separator'` • `headers` > `interchange-ctrl` > `attribute 'subelement-separator'` • `headers` > `interchange-ctrl` > `attribute 'segment-terminator'` • `headers` > `interchange-ctrl` > `attribute 'repetition-separator'` (only applies to X12 4020 or later versions) If the delimiter character you want to use is an ASCII printable character, simply assign a string value such as `"|"` to the corresponding attribute. If you want to use a special character as a delimiter, specify it in a hexadecimal format. For example, `0xA`, where `0x` is a fixed prefix and `A` is the hexadecimal representation of the ASCII line-feed character. If your EDI payload uses a different encoding such as UTF-8, you can use the hexadecimal format to specify a Unicode character as a delimiter. For example, specify `0x03A60` to use the Unicode character `Φ` as a delimiter. Note that the receiving trading partner should be prepared to receive a UTF-8 EDI payload for this to work correctly. Note that the delimiter string value you specify must equate to only a single character. Multiple characters are not allowed. The only exception is the `segment-terminator`, which allows one character plus an optional ASCII carriage-return and a line-feed character at the end. For example, a `segment-terminator` value of `~0xd0xa` uses `~` followed by `<CR>` and `<LF>` as a `segment-terminator`, which means each EDI segment is output on a separate line, making it more readable.
`accept-message-on-errors`	See Accept Message on Errors.

Element	Description
`reject-message-on-errors`	See Reject Message on Errors.
`validate`	See Validate.
`input-source-context`	See Input Source Context.

Elements in TranslateOutput

Element	Description
`edi-payload`	This element is returned with the translated message in the EDI format, starting with the X12 ISA header envelope and ending with the IEA trailer envelope. The control numbers are automatically generated and inserted in the EDI content. Note that the EDI content is Base64 encoded.
`validation-errors-present`	See Validation Errors Present.

`validation-errors > error`
`validation-errors > validation-error-report`
	See Validation Errors - Error and Error Report.

`tracking-info`	This element is returned with a string representing a unique transmission identifier occurring in the output EDI. This string includes control numbers used in the output EDI message (interchange/group/transaction-set control numbers). This value is useful for tracking purposes.
translation-status	See Translation Status.
`input-source-context`	See Input Source Context.

Sample TranslateOutput XML for Outbound EDI

This section provides a sample TranslateOutput XML file for outbound EDI.

```
<executeResponse xmlns="http://xmlns.oracle.com/cloud/adapter/ediAdapter/
EDI_Generate_REQUEST/types">
    <TranslateOutput xmlns="http://xmlns.oracle.com/b2b/edi/generic/
document">
        <edi-payload>SVNBKjAwKiAgICAgICAgICAqMDAqICAgI...E1fg==</edi-payload>
        <validation-errors-present>true</validation-errors-present>
        <validation-errors>
            <error>
                <code>B2B-01757</code>
                <category>data_codelist</category>
                <summary>Element BEG02 (element id 92) contains an invalid
identification code [00] that is not listed in the allowed codes for this
element (for example: PR, LS, DR, DS, UD, UE ...).</summary>
                <location>edi-xml-document &gt; transaction-data &gt;
segment-BEG &gt; BEG02 (element id 92) | segment position 2 (starting with
ST segment) | element position 2</location>
            </error>
            <validation-error-report>[1] Error code: B2B-01757 |
category: data_codelist | message: (Severe) Element BEG02 (element id
92) contains an invalid identification code [00] that is not listed in
the allowed codes for this element (for example: PR, LS, DR, DS, UD,
UE ...). | edi-xml-document &gt; transaction-data &gt; segment-BEG
&gt; BEG02 (element id 92) | segment position 2 (starting with ST
segment) | element position 2</validation-error-report>
        </validation-errors>
        <tracking-info>tc=1015;gc=1015;sn=Acme Foods;rc=Global
Chips;ic=000000015;</tracking-info>
        <translation-status>Error</translation-status>
    </TranslateOutput>
</executeResponse>
```

7

EDI Standards Reference

This section provides reference information about EDI standards.

Topics:

EDI Concepts

Electronic Data Interchange (EDI) standards such as X12 and EDIFACT predate XML standards. EDI data is not self-descriptive like XML; it was designed to be machine-readable and compact because networks had very low data bandwidth at the time of their inception. There was no scope to expand the message by adding extra tags or metadata.

To decipher EDI data, you must have a separate schema definition available. B2B for Oracle Integration provides X12 and EDIFACT standard schemas and hides some EDI native format intricacies by translating them to (or from) a canonical XML format. You design the backend integrations and largely work with the canonical format only. However, the logical structure and schema of the canonical XML format closely resembles the EDI structure of segments and elements. That is why you need knowledge of the EDI envelope and data segment structures. You may also need knowledge of the EDI schema if you and your trading partner mutually decide to customize the standard schemas.

This section provides EDI standard-specific reference information, such as envelope structures and code lists for specific elements.

There are standard envelopes used commonly within all X12 document types and within all UN/EDIFACT document types. The envelope structures are also briefly described in the following sections for each standard.

How to Access EDI Specifications

A specification for an X12 or EDIFACT document includes detailed information about the structure and definition of each data segment. Referring to these details, you can decipher the information conveyed in a given EDI document. In B2B for Oracle Integration, the specifications are visible in an interactive schema editor or as an implementation guide that you generate for offline viewing.

Here is how you access these details from your Oracle Integration instance:

- Create a B2B schema for the necessary document type. The schema editor shows an interactive schema for that document in which you can browse the structure and select and view elements, data types, code lists, and other details.

- Generate an implementation guide for the necessary document type from the B2B Schema page (this feature is not available for EDIFACT documents). You can generate an implementation guide for a standard schema or a customized schema.

EDI X12

EDI X12 defines the following envelope structure and envelope code lists.

- Envelope Structure
- Envelope Code Lists

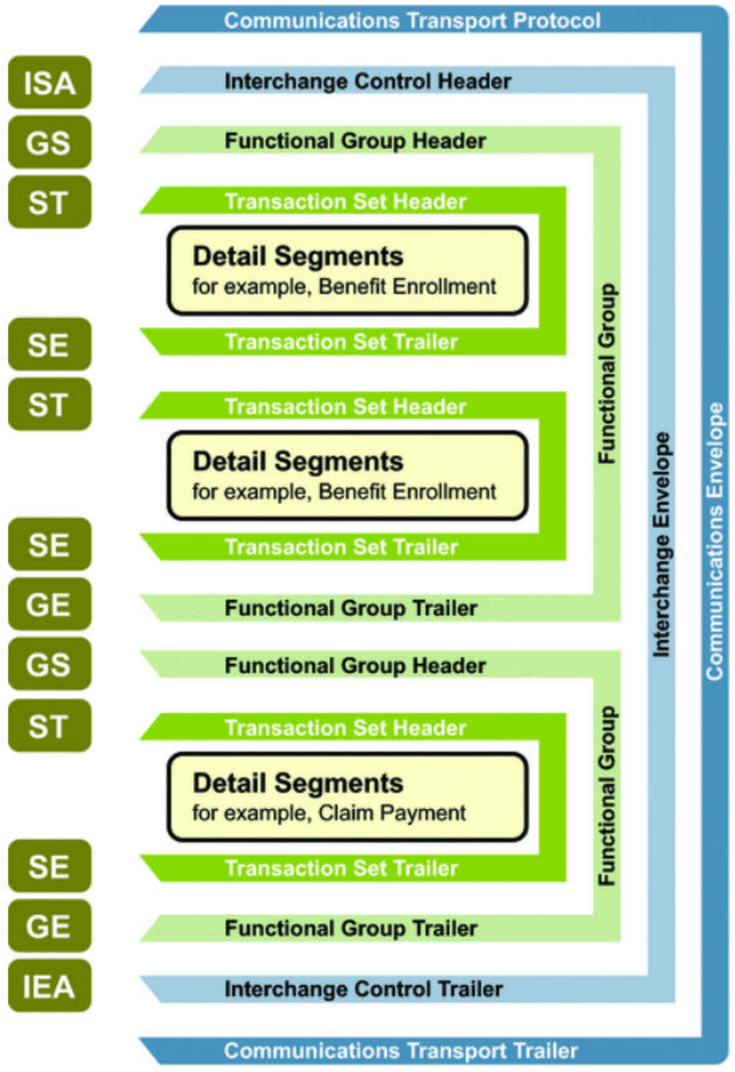
Envelope Structure

The rules for X12 envelope structure ensure the integrity of the data and the efficiency of the information exchange. The actual X12 message structure has primary levels that are hierarchical.

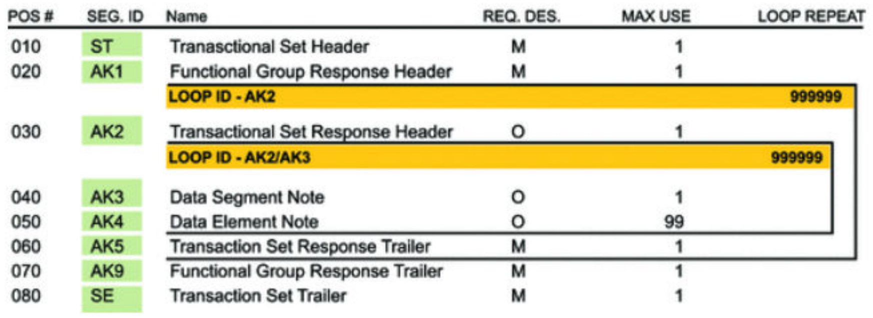
From highest to lowest, the structure is as follows:

- Interchange envelope
- Functional group
- Transaction set

The following diagram shows a schematic structure of X12 envelopes. Each of these levels is explained in further detail.



The following example shows the standard segment table for an X12 997 (Functional Acknowledgment) as it appears in the X12 standard and in most industry-specific implementation guides.



POS #	SEG. ID	Name	REQ. DES.	MAX USE	LOOP REPEAT
010	ST	Tranasctional Set Header	M	1	
020	AK1	Functional Group Response Header	M	1	
		LOOP ID - AK2			999999
030	AK2	Transactional Set Response Header	O	1	
		LOOP ID - AK2/AK3			999999
040	AK3	Data Segment Note	O	1	
050	AK4	Data Element Note	O	99	
060	AK5	Transaction Set Response Trailer	M	1	
070	AK9	Functional Group Response Trailer	M	1	
080	SE	Transaction Set Trailer	M	1	

Functional Groups (GS/GE)

Functional groups, often referred to as the inner envelope, consist of one or more transaction sets, all of the same type, which can be batched together into one transmission.

The functional group is defined by the header and trailer segments. The functional group header (designated GS) segment appears at the beginning and the functional group trailer (designated GE) segment appears at the end. Many transaction sets can be included in the functional group, but all local transactions must be of the same type.

Within the functional group, each transaction set is assigned a functional identifier code, which is the first data element of the header segment. The transaction sets that constitute a specific functional group are identified by this functional ID code.

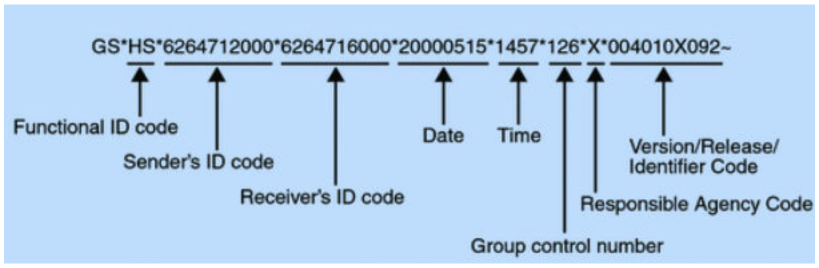
The GS segment contains:

- Functional ID code (the two-letter transaction code; for example, PO for an 850 Purchase Order, HS for a 270 Eligibility, Coverage, or Benefit Inquiry) to indicate the type of transaction in the functional group.
- Identification of the sender and receiver.
- Control information (the functional group control numbers in the header and trailer segments must be identical).
- Date and time.

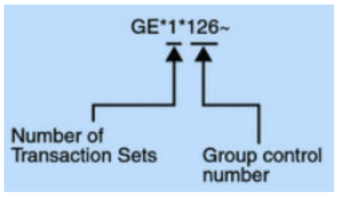
The GE segment contains:

- The number of transaction sets included.
- The group control number (originated and maintained by the sender).

The following example shows a functional group header (GS).



The following example shows a functional group trailer (GE).



Interchange Envelopes (ISA/IEA)

The interchange envelope, often called the outer envelope, is the wrapper for all the data to send in one transmission. It can contain multiple functional groups. This characteristic means that transactions of different types can be included in the interchange envelope, with each type of transaction stored in a separate functional group.

The interchange envelope is defined by the header and trailer; the interchange control header (designated ISA) appears at the beginning and the interchange control trailer (designated IEA) appears at the end.

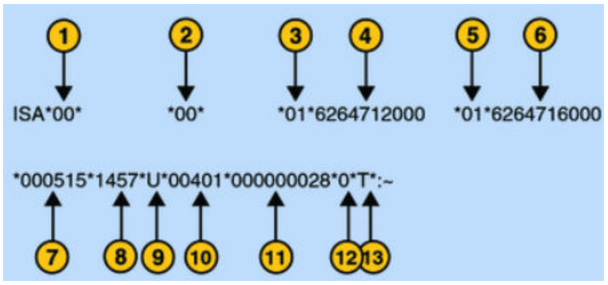
Besides enveloping one or more functional groups, the ISA and IEA segments include:

- Data element separators and data segment terminator
- Identification of sender and receiver
- Control information (used to verify the message was correctly received)
- Authorization and security information, if applicable

The following sequence of information is transmitted:

- ISA
- Optional interchange-related control segments
- Actual message information, grouped by transaction type into functional groups
- IEA

The following example shows an interchange header (ISA).

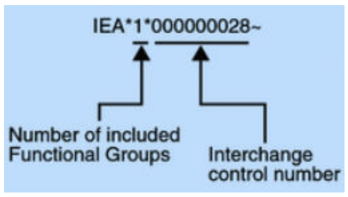


The following list describes the ISA segments:

1. Authorization information qualifier
2. Security information qualifier
3. Interchange ID qualifier
4. Interchange sender ID
5. Interchange ID qualifier
6. Interchange receiver ID
7. Date
8. Time
9. Repetition separator
10. Interchange control version number

11. Interchange control number
12. Acknowledgment requested
13. Usage indicator

The following example shows an interchange trailer (IEA).


IEA*1*000000028~
Number of included Functional Groups
Interchange control number

Transaction Sets (ST/SE)

Each transaction set (also known as a transaction) contains:

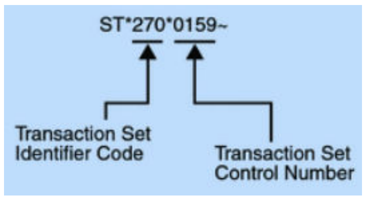
- Transaction set header (designated ST)
- Transaction set trailer (designated SE)
- Single message, enveloped within the header and footer

A transaction set has a three-digit code, a text title, and a two-letter code (for example, 997, Functional Acknowledgment (FA)).

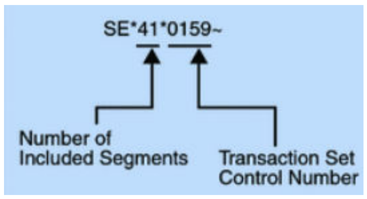
The transaction set is composed of logically-related pieces of information grouped into units called segments. For example, one segment used in the transaction set may convey the address: city, state, postal code, and other geographical information. A transaction set may contain multiple segments. For example, the address segment may be used repeatedly to convey multiple sets of address information.

The X12 standard defines the sequence of segments in the transaction set and also the sequence of elements within each segment. The relationship between segments and elements can be compared to the relationship between records and fields in a database environment.

The following example shows a transaction set header (ST).


ST*270*0159~
Transaction Set Identifier Code
Transaction Set Control Number

The following example shows a transaction set trailer (SE).



Envelope Code Lists

This reference section provides a list of allowed values in X12 envelope segments, ISA and GS, for those elements that must use values from an enumerated code list.

Authorization Information Qualifier (ISA01)

The following table shows the allowed values for the EDI interchange ID qualifier in X12, described in Schema Elements for Outbound EDI.

Value	Description
00 (default)	No Authorization Information Present (No Meaningful Information in I02)
01	UCS Communications ID
02	EDX Communications ID
03	Additional Data Identification
04	Rail Communications ID
05	Department of Defense (DoD) Communication Identifier
06	United States Federal Government Communication Identifier
07 **	Truck Communications ID
08 **	Ocean Communications ID

Note:

The highlighted code values are only valid for X12 versions 5020 and newer.

Security Information Qualifier (ISA03)

The following table shows the allowed values for the security information qualifier in X12, as described in Schema Elements for Outbound EDI.

Value	Description
00 (default)	No Security Information Present (No Meaningful Information in element I04, Security Information)
01	Password

EDI Interchange ID Qualifier (ISA05 and ISA07)

The following table shows the allowed values for the EDI Interchange ID Qualifier in X12, as described in Define the Host Profile and Schema Elements for Outbound EDI (headers > interchange-ctrl > sender-id-qualifier and receiver-id-qualifier).

Value	Description
01	Duns (Dun & Bradstreet)
02	SCAC (Standard Carrier Alpha Code)
03	FMC (Federal Maritime Commission)

Value	Description
04	IATA (International Air Transport Association)
07 **	Global Location Number (GLN) (A globally unique 13 digit code for the identification of a legal, functional, or physical location within the Uniform Code Council (UCC) and International Article Number Association (EAN) numbering system.)
08	UCC EDI Communications ID (Comm ID)
09	X.121 (CCITT)
10	Department of Defense (DoD) Activity Address Code
11	DEA (Drug Enforcement Administration)
12	Phone (Telephone Companies)
13	UCS Code (The UCS Code is a Code Used for UCS Transmissions; it includes the Area Code and Telephone Number of a Modem; it Does Not Include Punctuation, Blanks or Access Code.)
14	Duns Plus Suffix
15	Petroleum Accountants Society of Canada Company Code
16	Duns Number With 4-Character Suffix
17	American Bankers Association (ABA) Transit Routing Number (Including Check Digit, 9 Digit)
18	Association of American Railroads (AAR) Standard Distribution Code
19	EDI Council of Australia (EDICA) Communications ID Number (COMM ID)
20	Health Industry Number (HIN)
21	Integrated Postsecondary Education Data System, or (IPEDS)
22	Federal Interagency Commission on Education, or FICE
23	National Center for Education Statistics Common Core of Data 12-Digit Number for Pre-K-Grade 12 Institutes, or NCES
24	The College Board's Admission Testing Program 4-Digit Code of Postsecondary Institutes, or ATP
25	American College Testing Program 4-Digit Code of Postsecondary Institutions, or ACT
26	Statistics of Canada List of Postsecondary Institutions
27	Carrier Identification Number as assigned by Health Care Financing Administration (HCFA)
28	Fiscal Intermediary Identification Number as assigned by Health Care Financing Administration (HCFA)
29	Medicare Provider and Supplier Identification Number as assigned by Health Care Financing Administration (HCFA)
30	U.S. Federal Tax Identification Number

Value	Description
31	Jurisdiction Identification Number Plus 4 as assigned by the International Association of Industrial Accident Boards and Commissions (IAIABC)
32	U.S. Federal Employer Identification Number (FEIN)
33	National Association of Insurance Commissioners Company Code (NAIC)
34	Medicaid Provider and Supplier Identification Number as assigned by individual State Medicaid Agencies in conjunction with Health Care Financing Administration (HCFA)
35	Statistics Canada Canadian College Student Information System Institution Codes
36	Statistics Canada University Student Information System Institution Codes
37	Society of Property Information Compilers and Analysts NEW
38 **	The College Board and ACT, Inc. 6-Digit Code List of Secondary Institutions
AM	Association Mexicana del Codigo de Producto (AMECOP) Communication ID
NR	National Retail Merchants Association (NRMA) - Assigned
SA **	User Identification Number as assigned by the Safety and Fitness Electronic Records (SAFER) System
SN	Standard Address Number
ZZ	Mutually Defined

Note:

The highlighted code values are only valid for X12 versions 5020 and newer.

There is also no default value used. You must specify a value by adding a B2B identifier to the host profile or trading partner or mapping it in the outbound backend integration.

Interchange Control Standards Identifier (ISA11 - version 4012 and older)

The following table shows the allowed values for the Interchange Control Standards Identifier in X12. See Schema Elements for Outbound EDI.

Note that this element has been repurposed to specify a Repetition Separator in versions 4020 and newer.

Value	Description
U (default)	U.S. EDI Community of ASC X12, TDCC, and UCS

Acknowledgment Requested (ISA14)

The following table shows the allowed values for the Acknowledgment Requested in X12. See Schema Elements for Outbound EDI.

Value	Description
0 (default)	No Interchange Acknowledgment Requested
1	Interchange Acknowledgment Requested (TA1)

Usage Indicator (ISA15)

The following table shows the allowed values for the Usage Indicator in X12. See Schema Elements for Outbound EDI (headers > interchange-ctrl > usage-indicator).

Value	Description
I	Information
P (default)	Production Data
T	Test Data

Responsible Agency Code (GS07)

The following table shows the allowed values for the responsible agency code in X12. See Schema Elements for Outbound EDI (headers > group > resp-agency-code).

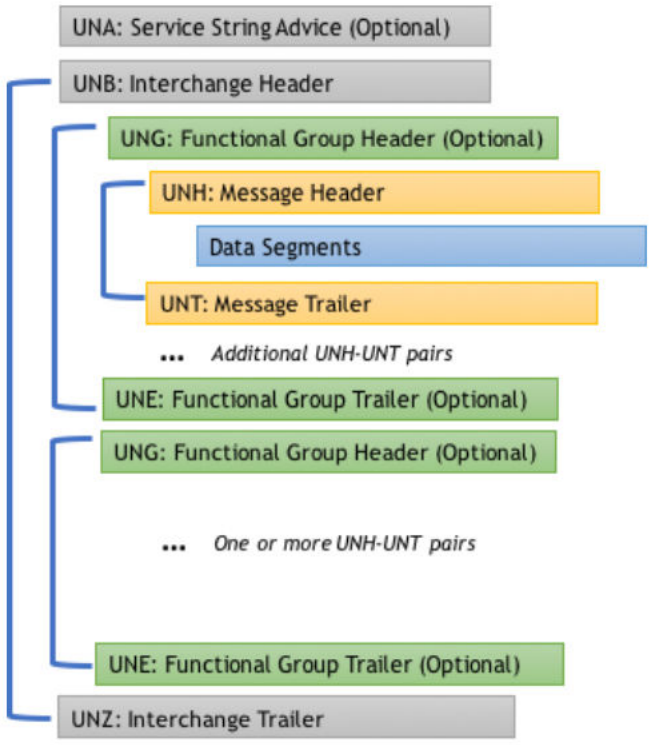
Value	Description
T	Transportation Data Coordinating Committee (TDCC)
X (default)	Accredited Standards Committee X12

EDI UN/EDIFACT

UN/EDIFACT defines the following envelope structure.

Envelope Structure

Full details are available. See Part 4. UN/EDIFACT Rules - Chapter 2.2 Syntax Rules - Annex B.


UNA: Service String Advice (Optional)
UNB: Interchange Header
UNG: Functional Group Header (Optional)
UNH: Message Header
Data Segments
UNT: Message Trailer
... Additional UNH-UNT pairs
UNE: Functional Group Trailer (Optional)
UNG: Functional Group Header (Optional)
... One or more UNH-UNT pairs
UNE: Functional Group Trailer (Optional)
UNZ: Interchange Trailer

EDIFACT envelopes are defined based on a syntax version. You can find details about each element and code lists for each syntax version below:

Version	Reference
Syntax Versions 1, 2, 3	https://www.gefeg.com/jswg/v3/data/v3_docs.htm
Syntax Versions 4	https://www.gefeg.com/jswg/v4/data/v4_docs.htm