



**eBook**

# **Connecting EDI to an API:**

## Bridging the Data Gap

Learn how the modern technology of an API takes outdated EDI to new heights.



## What is EDI?

Established in the late 1960's, Electronic Data Interchange (EDI) pioneered structured data transmission via computer— and many companies continue to use this method of data communication today. EDI provides a consistent and predictable way to transmit data between two computer systems. Over time, new EDI standards and new EDI transaction sets were released to accommodate industry-specific requirements (e.g., Automotive, Grocery, Apparel), a changing supply chain landscape (e.g., third party logistics [3PLs], load tendering, cargo tracking), and the idiosyncrasies of international business (e.g., EDIFACT, ZENGIN, TRADACOMS). These standards left room for interpretation between buyers and sellers, which led to organizations defining their own EDI specifications. This is why an EDI order from Target doesn't have the same contents as an EDI order from Macy's, Wayfair, Amazon, etc.

EDI suffers from many limitations: it can be difficult to understand/translate (a necessity for systems dedicated to processing the format) and can't support loosely structured/quickly evolving data needs. Nevertheless, many systems and companies still support this format today since it's safe, reliable, and suits many of their core business needs.

## What is an API?

Application Programming Interface (API) is the newest form of data communication. Utilizing real-time data exchange (rather than batch exchanges under standard EDI protocols), API provides data in several structured formats, such as JSON and XML. APIs and their data formats are being widely adopted by many companies to serve their complex data needs. APIs are typically well documented and present data in a human-readable format. This enables faster data mapping since it's possible to infer the meaning and usage of many fields directly from the data itself rather than having to decipher the meaning of different EDI segments and elements for each trading partner.

This modern approach to data transmission enables flexible, future-proofed, and cloud-based data transmission not only for basic business transactions but also frequently-changing data like inventory availability and more complex data structures required for rich product content.

**Both data formats support the same goal— structured data communication for computer processing.** So, if you're considering which one to support, you may be torn as to which best fits your business needs.

## Can I support both?

**Yes.** Whether your organization is on the buying or selling side of a business transaction, you'll likely need to support both EDI and API. While many large retailers and distributors mandate EDI, many online marketplaces and storefronts only support API. The same can be said from a buying perspective, as digitally native eCommerce brands expect API connectivity, while large, traditionally retail-focused manufacturers and distributors may prefer EDI. Being able to support a wide variety of external connections allows your organization to unlock new sources of revenue and supply.

Can I support both EDI and API?

**YES**



The backend, though, is a different story. Many software development teams prefer working with APIs for a few key reasons, particularly the ease with which data requirements can be understood through the data itself. Most EDI systems have thoroughly documented data requirements, but the learning curve associated with understanding those requirements presents a sizable barrier to entry. Supporting both API and EDI could be a large technical project, likely requiring a significant amount of code or multiple solution providers to manage trading partner specific communications and EDI mapping. On the other hand, it would enable data connectivity with both older and modern systems.

If you decide to partner with a company that does not provide an API, but rather uses structured EDI, how will you go about creating the link between an older, rigid data system and a modern, flexible one? And how will you handle the velocity required in today's fast-moving eCommerce world? **Enabling support for both EDI and API through a single solution allows you to get over these major hurdles.**

Below, we provide a breakdown of the major data similarities of two formats, using purchase orders from popular eCommerce retailers. Once you see the similarities between API and EDI data structures, you may want to explore creating your own integration to handle both legacy and modern data requirements through a single platform.



If you decide to partner with a company that does not provide an API, but rather uses structured EDI, **how will you go about creating the link between an older, rigid data system and a modern, flexible one?**

## Sending & Receiving Orders

The most complex part of mapping EDI data is understanding which elements are important and where they reside based on the trading partner's EDI specifications. These key elements can be used when mapping this data back into an API-friendly data structure. To demonstrate this, we will use samples of EDI orders from different organizations. For the API sample data, we will use examples from the [CommerceAPI](#), made available by Logicbroker.

Within a purchase order, there is a multitude of data that may be sent, but for our purposes, we'll focus on purchase order number, ship-to address, and line-level details (i.e., which products were sent and their quantity).

When sending/receiving EDI orders (referred to as an 850 in the X12 standard), this information can generally be found in **the following fields:**

- **PO Number** - BEG segment
- **Ship To Address** - N1, N2, N3, N4 fields
- **Line Level Information**
  - *Product Identifier* - PO1 segment (Any number of qualifiers)
  - *Quantity - PO1 Segment* (Typically Element number 2)

To those unfamiliar with EDI, you may immediately see an issue with the field naming: the segments are not clearly labeled (“PO1” vs “Line Information”), and the specific element the data will appear in is somewhat arbitrary. Because EDI is an older technology, from the early days of the Internet, keeping the data as small as possible was a high priority, since bandwidth was limited. This is responsible for many of the issues today’s technical teams have with building their own EDI integrations.

While the segments are poorly named, and the elements (where each data point lives) arbitrary, **these values can then be mapped to an API like so:**

- **PO Number** = Order.PartnerPO
- **Ship To Address** = Order.ShipToAddress
- **Line Level Information**
  - *Product Identifier* = Order.OrderItem.ItemIdentifier.
  - PartnerSKU

The [CommerceAPI](#) uses JSON as its standard format, which allows for programmatic use of objects to represent each of these relationships.

## Here is a sample order from the [CommerceAPI](#):

```
{
  "RequestedShipDate": "2020-01-02T00:00:00",
},
"TypeCode": "SA",
"SalesRequirement": 0,
"OrderLines": [
  {
    "ItemIdentifier": {
      "SupplierSKU": "1263673",
      "PartnerSKU": "1263673"
    },
    "Price": 33.49,
    "PriceCode": "TE",
    "Description": "KS PROTEIN BARS CHOC CHIP",
    "Discounts": [
      {
        "DiscountPercent": 0.0,
        "DiscountAmount": 0.0
      }
    ],
    "Taxes": [],
    "IsDropShip": false,
    "Quantity": 1,
    "QuantityUOM": "EA",
    "LineNumber": "01",
    "Weight": 5.0,
    "ExtendedAttributes": [
      {
        "Name": "PartnerLineID",
        "Value": "01"
      }
    ]
  }
],
"OrderNumber": "840490842",
"VendorNumber": "0002165100",
"PartnerPO": "00847002570589",
"OrderDate": "2020-01-02T00:00:00",
"Taxes": [],
"PaymentTerm": {},
"ShipmentInfos": [
  {
    "CarrierCode": "UPS",
    "ClassCode": "UPS-3D-2",
    "SenderClassCode": "UPS_3D",
    "ServiceLevelDescription": "UPS 3 Day
Select"
  }
],
"ShipToAddress": {
  "City": "DELRAY BEACH",
  "State": "FL",
  "Country": "US",
  "Zip": "33445-5308",
  "AddressCode": "00011",

```

*[cont. on next page]*

```

},
"BillToAddress": {
  "City": "DELRAY BEACH",
  "State": "FL",
  "Country": "US",
  "Zip": "33445",
  "Province": "",
  "StateCode": "",
},
"ExtendedAttributes": [
  {
    "Name": "BEG01_Purpose",
    "Value": "00"
  },
  {
    "Name": "REF - Internal Control Number",
    "Value": "815837656"
  }
],
"TotalAmount": 33.49,
"HandlingAmount": 0.0,
"DropshipAmount": 0.0,
"Note": "Custom Note",
"SenderCompanyId": 100614,
"ReceiverCompanyId": 126739,
"Identifier": {
  "SourceKey": "840490842",
  "LogicbrokerKey": "35978387",
  "LinkKey": "200116260198"
},
"DocumentDate": "2020-01-02T22:20:49.57",
"StatusCode": 100
}

```

## A Real-Life Example

Here, we'll look at the way three retailers send order information via EDI:

### Macy's - EDI 850 (Purchase Order)

ST	ST*850*0001
BEG	BEG*00*SA*64788163**20200102
CUR	CUR*X6*USD
REF	REF*X9*815836841
REF	REF*IA*0000017176
REF	REF*CO*1888489733
REF	REF*CNO*187971658
REF	REF*11*CHSHOESCREW
SAC.SAC	SAC*C*H850***427*****TAX TOTAL
ITD	ITD*08**0**0**30*00
DTM	DTM*001*20200202
TD5	TD5****UPS_N.CG
CTB	CTB*OR*IF YOU HAVE ANY PROBLEMS WITH THIS SHIP
N9.N9	N9*SD*11
N1.N1	N1*BT*SILVA
N1.N3	N3*11432
N1.N4	N4*LAKE VIEW TERRACE*CA*91342-6503*US
N1.N1	N1*ST*SILVA
N1.N3	N3*11432
N1.N4	N4*LAKE VIEW TERRACE*CA*91342-6503*US
N1.N1	N1*SF**93*NV
N1.N1	N1*PO**93*0149
PO1.PO1	PO1*01*1*EA*28*TE*****UP*885999115139

### Wayfair - EDI 850 (Purchase Order)

ST	ST*850*879507
BEG	BEG*00*DS*CS205469363**20200102
REF	REF*VR*19010*75126
REF	REF*2P*Wayfair
PER	PER*IC*Del Hutson Designs*FX*4697286012*TE*214336488
CTP	CTP**WHL
ITD	ITD*01*****3% 10 Net 60
DTM	DTM*010*20200107
TD5	TD5**2*FDEG*ZZ*FDHD
N9.N9	N9*TH*346593300
N1.N1	N1*BT*Wayfair
N1.N3	N3*4
N1.N4	N4*Boston*MA*02116*US
N1.PER	PER*BJ*BT*TE*6175326815*FX*617-502-7798*EM*Wayfair
N1.N1	N1*ST*Robin
N1.N3	N3*7226
N1.N4	N4*L...*OH*43537*US*ZN*R
N1.PER	PER*DC*ST*TE*4194500603***EM*customeremail@wayfair
N1.N1	N1*LW*L
N1.N3	N3*7226
N1.N4	N4*...*OH*43537*US
PO1.PO1	PO1*1*1*EA*29*WH*VN*DHD3118wh
PO1.PID.PID	PID*F***Treyvon Industrial Grace Wall ShelfSize: 1 75 H x 24

To view all available fields, [view the CommerceAPI swagger documentation here.](#)

## Amazon - EDI 850 (Purchase Order)

ST	ST*850*0001
BEG	BEG*00*NE*8542VPSA*20191230
REF	REF*CR*WONV5
CSH	CSH*N
DTM	DTM*064*20191230
DTM	DTM*063*20200108
N1.N1	N1*ST**92*OAK3
PO1.PO1	PO1*1*3*EA*146.99*PE*UP*811225030112
CTT.CTT	CTT*1*3
SE	SE*10*0001

Through these three examples, you can see there is a high degree of variance between the fields retailers may or may not send on their EDI purchase orders. A generally adopted standard is putting the purchase order number in the third element of the BEG segment. However, this is still difficult to infer without having experience working with EDI purchase orders and does not apply to international EDI standards or industry specific sub-standards. Looking at the PO1 segment, we see the following from each retailer:

### Macy's:

P01\*01\*1\*EA\*28\*TE\*\*\*\*\*UP\*885999115139

Macy's uses UPC codes as a product identifier, and this can be determined by their use of the UP qualifier before the UPC code. **But compare to Amazon's item identifier usage:**

### Amazon:

P01\*1\*3\*EA\*146.99\*PE\*UP\*811225030112

Both use UPC as a product identifier and both use a UPC qualifier. However, they send the value in a different position, meaning that a specific map would need to be setup for each retailer that can identify where this product identifier will be sent, depending on who is trading.



**Logicbroker** uses EDI mapping technology to map "UP" EDI qualifiers to the UPC field, and **supports vendors who send more than one qualifier via EDI.**

As can be seen above, Wayfair does not send UPC codes, and uses a "VN" (vendor number) qualifier to identify a different type of product identifier completely.

When leveraging an API, these differences can be minimized by using clear line-level specific names. Take for instance how these fields are mapped in Logicbroker using the [CommerceAPI](#):

```

"OrderLines": [
  {
    "ItemIdentifier": {
      "SupplierSKU": "1263673",
      "PartnerSKU": "1263673",
      "UPC": "888126367311"
    },
  },

```

**Logicbroker** uses EDI mapping technology to map "UP" EDI qualifiers to the UPC field, and supports vendors who send more than one qualifier via EDI, i.e. sending a "VN," in addition to a "UP" qualifier would mean that both the "SupplierSKU" and "UPC" item level fields are mapped as needed.

### Wayfair: Shipping Address Information

N1\*BT\*Wayfair  
 N3\*4 Copley Place, Floor 7  
 N4\*Boston\*MA\*02116\*US  
 PER\*BJ\*BT\*TE\*6175326815\*FX\*617-502-7798\*EM\*WayfairOps5@wayfair.com  
 N1\*ST\*First Last  
 N3\*123 Address Way  
 N4\*City\*0H\*43537\*US\*ZN\*R  
 PER\*DC\*ST\*TE\*4195550603\*\*\*EM\*customeremail@wayfair.com  
 N1\*LW\*First Last  
 N3\*123 Address Way  
 N4\*City\*0H\*43537\*US

A final example of the variances on EDI 850 Purchase Orders would be the N1-N4 segments, which describe address information. Taking a look at the examples from above:

### Macy's: Shipping Address Information

N1\*BT\*First Last~  
 N3\*123 Address Way~  
 N4\*City\*CA\*91342-6503\*US~  
 N1\*ST\*First Last~  
 N3\*123 Address Way~  
 N4\*City\*CA\*91342-6503\*US~  
 N1\*SF\*\*93\*NV~  
 N1\*P0\*\*93\*0149~

Note the additional "N1\*SF" and "N1\*PO" segments indicating special instructions for the vendor and the way they will ship.

Note how Wayfair includes information to contact the customer in the "PER" segment. Additionally, there is an N1/Address loop that uses an "LW" qualifier. LW is another way to identify the customer, but if you had never seen it before, you may not know what it represents.

### Amazon: Shipping Address Information

N1\*ST\*\*92\*0AK3

Amazon sends a simple address code for where products need to be sent.

**EDI retailers may use the following qualifiers to identify different address types:** Bill To Address (BT), Ship From Address (SF), and Ship To Address (ST). However, we see in the above examples that Amazon sends their address using an address code (a code that represents the entire address), another variance that would need to be accounted for in an address/EDI map.

Logicbroker's API handles these variances by having a clearly labeled object for each address type, as well as allowing for specific address codes within each address, which can be mapped later during the order lifecycle.

```

"ShipToAddress": {
  "CompanyName": "string",
  "FirstName": "string",
  "LastName": "string",
  "Title": "string",
  "Address1": "string",
  "Address2": "string",
  "City": "string",
  "State": "string",
  "Country": "string",
  "Zip": "string",
  "Province": "string",
  "AddressCode": "string",
  "StateCode": "string",
  "CountryCode": "string",
  "Phone": "string",
  "ContactID": "string",
  "ContactType": 0,
  "Email": "string",
  "TaxNumber": "string",
  "FaxNumber": "string",
  "Note": "string",
  "ExtendedAttributes": [
    {
      "Name": "string",
      "Value": "string",
      "Section": "string"
    }
  ]
},

```

```

"BillToAddress": {
  "CompanyName": "string",
  "FirstName": "string",
  "LastName": "string",
  "Title": "string",
  "Address1": "string",
  "Address2": "string",
  "City": "string",
  "State": "string",
  "Country": "string",
  "Zip": "string",
  "Province": "string",
  "AddressCode": "string",
  "StateCode": "string",
  "CountryCode": "string",
  "Phone": "string",
  "ContactID": "string",
  "ContactType": 0,
  "Email": "string",
  "TaxNumber": "string",
  "FaxNumber": "string",
  "Note": "string",
  "ExtendedAttributes": [
    {
      "Name": "string",
      "Value": "string",
      "Section": "string"
    }
  ]
},

```

Although we have touched on three commonly used EDI segments, retailers may send entirely different data in completely different ways (e.g., REF segments, DTM, etc.) A key takeaway regarding EDI is that each piece of information may live in a different data point, but it is up to your EDI system/provider to map these values in a way that your system can handle.

APIs provide a human-readable format to this data, which allows for quicker system mapping, greatly reducing the amount of time it takes to onboard and transact data with your partners. To learn more, check out this [additional information about channel integrations via API](#). 