

Channel Based Architecture (CBA)

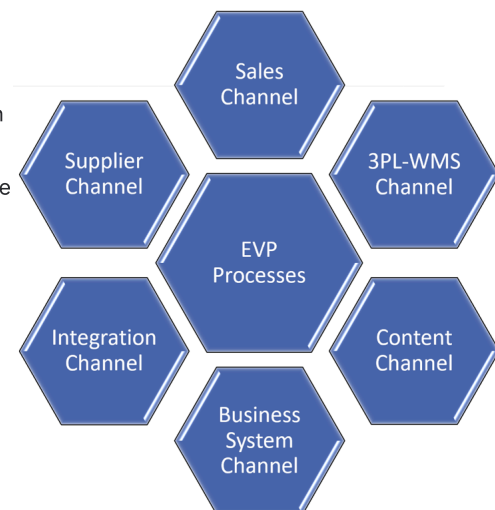
TABLE OF CONTENT

- [EVP Channel Process Model Types](#)
- [EVP Integration Methodology](#)
- [EVP Business Object Model](#)
 - [EVP Verbs](#)
 - [EVP Nouns](#)
 - [Typical Message Formats](#)
 - [File-Based Configuration](#)
 - [EDI Configuration Framework](#)
- [Flexible Process and Deployment Models](#)



Etail Solutions' approach to the challenging problem of today's Digital Commerce hyper-integration needs is to implement the interfaces between EVP and other systems such as Business Systems (ERP, CRM, POS, etc.) websites, marketplaces, content providers, and suppliers to leverage a loosely coupled channel-driven (LCCD) model, which allows:

- Transaction data will be utilized within predefined but configurable domain-centric processes, enabling the orchestration and Synchronization of business strategies. This can be done while isolating each connection point from the complexity of data formatting and, more importantly, eliminating the complexity of enabling business rules or event-driven escalation in an EDI point-to-point or Enterprise Service Bus message integration model.
- Creation of business process instrumentation to track the flow of work through the application.
- A loosely coupled channel-driven architecture positions a business to grow and minimizes the impact as technologies change. The architecture enables the business to add new suppliers, sales channels, etc., or swap out channels with minimal impact on the other parts of the system. It also readily integrates acquisitions, connects to B2B (business-to-business) partners, and connects to the customer.
- The approach to deliver this needed flexibility and extensibility is a loosely coupled application architecture.




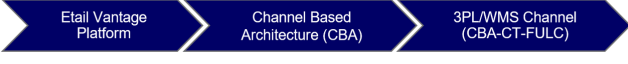



EVP Channel Process Model Types

Channels and Adaptor programs characterize loosely coupled solutions. Adaptors implement flexibility and insulate applications and business processes from one another and the details of data transport.

The core channel types supported within EVP are:

Process Model Type	Purpose
<pre>graph LR; A[Etail Vantage Platform] --> B[Channel Based Architecture (CBA)] --> C[Sales Channel (CBA-CT-SALC)]</pre>	It supports integration with external sales channels such as marketplaces, websites, POS, etc., where an independent seller of branded or non-branded products actively sells their Catalog of products. The standard Sales Channel process model supports publishing or linking to a sales channel listing and publishing inventory and price based on the current available inventory, location, and estimated shipping cost. Once an order is placed, it must be fulfilled at the best cost based on the service level associated with the sales order.
<pre>graph LR; A[Etail Vantage Platform] --> B[Channel Based Architecture (CBA)] --> C[Supplier Channel (CBA-CT-SUPC)]</pre>	It supports integration with suppliers to support warehouse replenishment, sales order-driven cross-docking of pull/flow-based purchase orders, and drop shipping. The Supplier Channel process model supports integration with a supplier to receive various master data forms, including Item catalog details, images, pricing, MAP/UPP, or point-in-time inventory feeds that could be fully updated daily or delta changes in the past hour. This supplier data controls what is available for sale on Cross Docking or Drop Ship order fulfillment models, where the items will be ordered on a Purchase Order once sold. The adaptor will also support one or more touch points associated with PO processing, i.e., sending PO to the supplier, receiving shipment details back from the supplier, etc.

	It is used to support event-driven or ad-hoc external system integration that needs to be called to enable the execution of a required process; the most frequent implementation is used for shipment manifesting rate shopping of a sales order and generation of carrier package labels.
	It supports integration with a business system such as ERP, POS, CRM, Financial, etc. It can support a single transaction, such as feeding fulfilled sales order details to a financial system, or multiple transactions to control the selling of inventory controlled by the business system and coordinate its fulfillment.
	It integrates with a rich content source for sold items, such as Enterprise PIM, Etilize, Ingram Book Catalog, Content Café, etc. Matching the item to this rich content uses UPS, EAN, ISBN, MPN, or a Distributor's Item Number.
	It supports integration with a subsystem responsible for tracking the bin-level location and inventory quantity owned by the customer. It also handles the receipt against an inbound order and its shipment against an outbound order without creating a purchase order.
	The EVP API provides an integration model to complement other channels, support customer-specific scenarios. or deploy EVP in a headless model where another system

EVP Integration Methodology

As discussed in the prior sections, there are almost unlimited combinations of how we can connect to an external system, some predefined formats such as EDI or a sales channel API; for other channels, it can be configured at the processor level. Therefore, it is challenging to answer the question, “Send me your standard format accurately, and we will work to that!” To provide a clearer understanding of what is possible on a specific client or system integration project, we will provide details on the methodology of how we implement our channel-centric model in the digital commerce space.

At Etail Solutions, we straightforwardly approach this complex problem based on our confidence in our channel-centric Architecture and EVP-enabled processes. Every discussion on this topic is focused on answering these questions:

- What is the business requirement (or problem) we are trying to enable, and what are the associated processes we need to enable?
- Does the process need to flow across multiple systems (channels) within an organization or across different organizations to support the business requirement? As a SaaS deployment model, everything we do is, by definition, inter-enterprise integration, but it is very important.
- To enable an intra-system or inter-system process, what business documents (Sales Order, PO, etc.) and associated synchronization events (message to communicate ASN Created, ASN Receipt Occurred, etc.) are required?
- What data elements are needed within a business document to enable standard workflow(s)
- What data elements are needed within a business document to enable exception workflow(s)
- Is a business requirement gap an integration issue (missing business document, event, or data elements) or a process issue (having the data but no application workflow or feature that can use it)? i.e., we need to make sure we are handling the gap in the right place

Once you understand the above, you can drill into the specific systems integrations required, including connection protocols and message formats. However, even here, the focus is on being pragmatic and understanding the need to accommodate the fact that we (Etail or Client) have limited ability to dictate the required process and associated integration. For example, the wait on the support line would be very long if you were calling Amazon or Walmart to get them to change their systems to work with your process and message formats!

Therefore, we typically start this topic with the following questions:

- Do you have access to the required business document data before making human-driven adjustments?
 - A spreadsheet with three users editing different tabs is not the start of a repeatable or scalable integration model; it may be phase 0 or used to handle an exception scenario, but it cannot be the foundation for the future.
- Do you have existing integration points that we can emulate? For example, an ERP system can talk to an external WMS; therefore, we can model EVP as a WMS and get the required data via standard WMS touchpoints, emulate being an external supplier to receive an EDI document, etc.
- Suppose the answer to the above is ‘NO.’ In that case, the next question is the easiest way for the client, supplier, or third-party technology provider to support the required flows: Let’s adopt an approach that they are familiar with and have the skills to implement.
- Only then would Etail Solutions say let’s work to our message format or connection technology, designed from the ground up to be flexible and support many different options. This approach has been driven by over 30 years of Etail leadership involvement in building integrated solutions across many business domains and working with standard organizations, such as the Open Application Group (OAGi) with its powerful canonical object messaging standard (OAGIS) or the Supply Chain Operational Reference model ([SCOR](#)) that was created by the Supply-Chain Council and is now part of the APICS organization

and is considered the cross-industry de facto standard strategy, performance management, and process improvement diagnostic tool for the supply chain management.

Ultimately, deploying an integrated solution requires detailed knowledge of the domain space, processes, and actions required to support the business solution. Etail Solution has this knowledge in the Digital Commerce domain. We have found the quickest time to value. User adoption is to allow our clients to stay within the technology stack they are familiar with and let Etail do the heavy lifting on how to conform to what is already in place, i.e., CSV files FTP'd from an old Unix system to Enterprise Service Bus or even open-source solutions such as RabbitMQ.

EVP's channel-level model enables domain-specific processing. Each model has one or more process adapters that support a set of predefined touchpoints that use a Connection Services layer that allows messages to be handled in a data format and connection protocol-neutral manner using a library of predefined services. The connection service model provides the following process isolation capabilities:

- **Application/Transportation Layer** is the element of the connection service used to enable a reliable and secure connection with a remote system and controls which network to connect to, where to find the required data elements, and how to handle remote deletion or archiving of the message. Supported options include:
 - **FTP** – File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and a server on a computer network
 - **TCP/IP**—Transmission Control Protocol (TCP) and Internet Protocol (IP) sockets provide a low-level connection for passing data packets. They are the foundation for the Internet Suite of protocols and are also used to communicate with material handling subsystems that need real-time interaction.
 - **SFTP** – SSH File Transfer Protocol (or Secure File Transfer Protocol) is a network protocol that provides file access, file transfer, and file management over any reliable data stream.
 - **AS2** – ‘Applicability Statement 2’ is a specification about how to transport structured business-to-business data securely and reliably over the Internet. Digital certificates and encryption enable a secure connection.
 - **HTTP**—The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example, by clicking a mouse or tapping the screen in a web browser.
 - **HTEVP** - HyperText Transfer Protocol Secure (HTEVP) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, including the Internet. In HTEVP, the communication protocol is encrypted using Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer (SSL). The protocol is, therefore, also often referred to as HTTP over TLS or HTTP over SSL.
 - **SMTP** - Simple Mail Transfer Protocol) is a communication protocol for electronic mail transmission.
 - **SOAP** – Simple Object Access Protocol is a messaging protocol specification for exchanging structured information to implement web services in computer networks.
 - **Others** – can implement client, supplier, or external system-specific application/transportation layers as needed.
- **Message Structure** – Use the message structure to define the type of payload sent between applications; it can be a single record per line, a multi-segment EDI document, or a node-centric XML structure. There needs to be a predefined message structure that has been mutually agreed upon between integration endpoints. This will often be standard-driven formats such as EDI X12 for EDI documents and Open Application Group Integration Schema for XML messages. It could also be a de facto standard based on widespread industry adoption (IDOCs from SAP) or a specific API/message specification format implemented by a supplier or sales channel.
 - This agreement represents the syntax of the inter-connection, meaning we are talking English, French, or Greek.
 - Optimization can occur in a message structure, enabling the highest processing speed possible. A system could send a full update sales order message, and the receiving system would need to process all order elements to understand the change in the business document and perform one or more updates or actions. A better solution is to send a streamlined message with only the attributes associated with the event details in the message structure, i.e., SendShipUnit related to a shipment occurring on a sales order.
 - Synchronous or asynchronous messages are possible. The Adapter will wait (Synchronously) until a response is received to the call before core processes continue, i.e., lookup items rich content on a content management system. But if 100K items need to be processed, a better solution would be to send a request for information (asynchronous) and continue, and then have another adapter look for the response in a folder or inbound message queue.

Possible formats include:

- **CSV** - A comma-separated values file is a delimited text file that uses a comma to separate values. Each line of the file is a data record.
- **TXT** - A text file (sometimes spelled text file; an old alternative name is a flat file) is a kind of computer file structured as a sequence of lines of electronic text.
- **Microsoft Excel** (.XLS or .XLSX) – A format used to store or transfer spreadsheets between users, but has several challenges when used for integration:
 - If a multi-tab spreadsheet, which tab should be processed
 - “10 + Sheet1: Cell A1” in a cell formula will not work. It needs to be the result in the cell.
- **EDI** - Electronic data interchange (EDI) is the concept of businesses electronically communicating information traditionally delivered on paper, such as purchase orders and invoices. EDI technical standards exist to facilitate parties transacting such instruments without making special arrangements. EVP has standard maps for v4010 and v5010, but can accommodate custom maps as needed

- **XML** - Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a human-readable and machine-readable format.
- **WSDL** – Web Services Description Language is an XML-based interface description language used to describe the functionality offered by a web service.
- **JSON** - JavaScript Object Notation is an open-standard file format or data interchange format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types (or any other serializable value).

Within a single channel, it is possible to leverage protocol and message format on a per touchpoint basis or even the same Touchpoint based on need, i.e., use a .csv file via SFTP to process a sizeable daily inventory file but then call a JSON-based RESTful API to lookup delta changes to inventory that have occurred during the last ‘n’ minutes. The overall integration model allows the scaling and distribution of processors combined with the ability to configure them as single or multi-threaded deployments to support scaling where various sales, suppliers, and integration channels process millions of events/messages per hour across our SaaS environment.

This loosely-coupled channel-driven architecture delivers multiple benefits to both Etail and its Clients:

1. Business solution flexibility. The impact of a change in one application/system extends only to its adapters.
2. Technology swap-out capabilities. Placing information management with its data source enables technology to swap out. The information management is contained in a program called an adapter. The adapter programs are placed at the “edge” of each participating application. An alternate way to think about adapter placement is in the “space” between the applications and the middleware transport. Placing the adapter programs in this “space” allows the concept of technology changes to extend to the middleware itself.
3. It enables hiding data structure details and application logic from other applications, also called encapsulation.
4. Common message objects that are defined in a dictionary.
5. Messages that specify the data formats for information exchange between applications are effectively application protocols. An application protocol allows application gateways to be implemented, and gateways change protocols.
6. Stable information flow between applications. A message is a contract. Once published, its format is static. Other systems can depend upon it. When a new data combination is needed, a new message is created. Only those applications interested in the latest data combination are affected. The impact is isolated to the new adapters.
7. Scalability - has been mentioned previously as the ability to add instances of an application “behind” the adapter. These instances can either share the workload or divide the workload. In shared workload implementations, each application instance is the same and does the same work. The inbound workload is distributed. Each application instance handles a predefined type of work in divided workload situations. An example of a predefined type of work is treating customers by geography. One system instance handles customers in the West, while another handles customers in the East. There are many other schemes for dividing the workload. Scalability is the ability to run multiple adapter copies for any single application.

EVP Business Object Model

Within each channel model implementation, a collection of processors enables the sending or receiving of business objects that orchestrate the channel-centric business process. These processors perform an Action on a Business Object that is represented as a Verb+Noun combination:

- The **Verb** is the action to be applied to the business object (the **Noun**). Examples of Verbs include Cancel, Add, Process, and Synchronize. Any additional information exclusively related to the action is stored with the Verb.
- The **Noun** is the business object or document that is being acted upon. Examples include Purchase Orders, Sales Orders, and Invoices.

Different types of verbs or actions can be performed on a Purchase Order; as such, the base Noun (e.g., Purchase Order) contains all of the information that might be present on a Purchase Order. The instantiation of each possible verb and noun combination further defines what must be provided to perform the intended transaction. For example, in a ProcessPurchaseOrder transaction, business partners and line item data must be provided, whereas in a **CancelPurchaseOrder** transaction, only the order identifier needs to be provided. Many business-level objects can be exposed to the core EVP processes and utilized as needed within the Integration layer.

EVP Verbs

Initialization Verb	Responding Verb	Purpose	Example
Process	Acknowledge	The PROCESS verb is used to request the processing of the associated business object by the receiving application or business partner to enable an overall business scenario that could trigger a business-level response to ACKNOWLEDGE the business object with either positive or exception details, which could trigger the sending of a CHANGE or CANCEL message.	Request: Receive_PurchaseOrder Response: AcknowledgePurchaseOrder

		The PROCESS business document is considered to be a legally binding message. For example, suppose a ProcessPurchaseOrder message is sent to a supplier, and the supplier acknowledges acceptance. The supplement must fulfill the purchase order unless an exception or change message is exchanged to adjust the agreement.	
Change	ChangeAcknowledge	<p>The CHANGE verb communicates a change in a previously sent PROCESS message.</p> <p>Includes options to indicate Add, Change, Delete, or Replace</p>	<p>Request: ChangePurchaseOrder</p> <p>Response: ChangeAcknowledgePurchaseOrder</p>
Cancel	CancelAcknowledge	The CANCEL verb communicates a change in a previously sent PROCESS message.	<p>Request: CancelPurchaseOrder</p> <p>Response: CancelAcknowledgePurchaseOrder</p>
Send	SendResponse	<p>The SEND verb communicates business object details to synchronize two systems. The SENDRESPONSE is optional.</p> <p>Includes options to indicate Add, Change, Delete, or Replace</p>	<p>Request: SendCustomerAddress</p> <p>Response: SendResponseCustomerAddress</p>
Get	Show	<p>The GET / SHOW Verb combination is mainly used in API scenarios when data is being requested and then returned.</p> <p>The purpose of the GET verb is to communicate to a business software component a request for an existing piece of information to be returned. The response to this request is the SHOW verb. The GET can request a single piece of information by using that information's primary retrieval field or key field, i.e., PO Number. It can also be used to request several instances of business objects, i.e., lists of open purchase orders.</p> <p>The SHOW verb is used when sending information about a specific business document or entity instance. The SHOW verb may be used:</p> <ol style="list-style-type: none"> 1. To respond to a GET request or 2. In a push notification to another application based on a business event. <p>In the published scenario, Business Objects based on this verb do not commonly cause updates; sometimes, the component receiving the SHOW decides to use the information it receives to update. This is entirely the decision of the receiving software component and is not forbidden</p>	<p>Request: GetPurchaseOrder</p> <p>Response: ShowPurchaseOrder</p>

EVP Nouns

Noun		Related Verbs	Detailed Description
Catalog		<p>Get / Show</p> <p>Send</p>	The Catalog business object is a list of items or commodities. The items may be arranged according to a classification scheme. The Catalog can identify its classification scheme and the classifications and features defined within the catalog. Within the Catalog, each item can be classified into one or more categories, and the specifications of each item can be cataloged. A Catalog has at least one publisher and one or many suppliers for the items in the Catalog. The CatalogHeader describes the Catalog as a whole and communicates information that pertains to the Catalog. The CatalogLine describes what is in the Catalog and can include:

		<ul style="list-style-type: none"> Item(s) Commodity Class Catalog—This identifies the classification scheme used by Catalog items. It also identifies all the classifications within this scheme and their hierarchical representations. The catalog features defined in this classification scheme and the features associated with each hierarchical classification representation can also be specified. UOMCode SupplierParties ItemPrice <p>The GetCatalog aims to enable a business application module or system to request catalog information. The Catalog information that is requested by the GetCatalog may include · Item Identifiers · Specifications · Pricing Information agreed on either · Purchase Agreements · Price List · Availability and Delivery Information · Related Items, and accessories. Many possible business applications in several environments may use this capability. Some examples of usage scenarios are · Manufacturer exchanging catalogs with distributors/ suppliers/e-marketplaces · Distributors/ Suppliers/ e-marketplaces exchanging catalogs with Buyers or other trading partners. It may also be necessary to support Content Supplier Management (CSM) scenarios. In this scenario, a company will provide a service for sourcing and codifying many companies' products and publishing a consolidated catalog.</p> <p>The purpose of the SendCatalog is to synchronize catalog information across systems. The Send indicates that the sender of the SendCatalog is the owner of the information being passed. Additionally, it is understood that the receiver of the SendCatalog is a sub-system that must accept the data provided by the owner of the information.</p>
ItemMaster	<div>Receive</div> <div>Get / Show</div> <div>Send</div>	<p>The ItemMaster business object represents any unique purchased part or manufactured product. Item, as used here, refers to the basic information about an item, including its attributes, cost, and locations. It does not include item quantities. Compare this to the noun InventoryBalance, which includes all quantities and other location-specific information. ItemMaster is used as the Item Master. The ItemMaster provides details about an Item in many domains of an enterprise. Each domain or functional area is interested in different views of the Item. The ItemMaster provides information on the following partial functional areas: Listfacturing and Order Management, the Catalog view, Logistics, Engineering, EngineeringDocument, CRM, and Financials.</p> <p>The purpose of ReceiveItemMaster is to provide Item information for goods or services to another business application module. The sending system may also initiate this message upon some business event occurring, and it is intended to request that the receiving system process and perform the required actions based on the content of the message.</p> <p>The GetItemMaster aims to enable a business application module to request information concerning a specific ITEM from another business application. The reply to this BOD is the ShowItemMaster. Many possible business applications in several environments may use this capability. For example, ERP, Inventory, or Manufacturing business application could use this to request item information.</p> <p>SendItemMaster provides Item information for goods or services to another business application module, enabling peer-to-peer systems to synchronize their item information. The sending system may also initiate this message upon some event occurring.</p>
Inventory	<div>Receive</div> <div>Change</div> <div>Get / Show</div> <div>Send</div>	<p>The Inventory business object includes all stocked items and primarily represents the quantities of each item by location. Other item-by-location information, such as serial or lot numbers, can also be included. The use of this noun does not include basic item master data that is independent of location, such as item description and dimensions. All Quantity information will be updated by replacing the entire field, not by incrementing the quantity.</p> <p>The purpose of the ReceiveInventory is to communicate the need to process InventoryBalance data that exists on separate Inventory systems, where the two systems are peers. This data is not the master data that describes the item's attributes, such as dimensions, weight, or unit of measure. The data describes the ITEM as it exists at a specific location.</p> <p>The purpose of the ChangeInventoryBalance is to enable the communication of a Change in the Inventory balance data on separate Inventory databases.</p>

		<p>The GetInventory aims to enable an application to request specific Inventory information from another business application module. The response to the GetInventory is the ShowInventory.</p> <p>The SendInventory enables the Synchronization of Inventory data on separate Item Master databases.</p>
InventoryCount	Receive	<p>The InventoryCount business object represents the results of a physical inventory or cycle count of the actual on-hand quantities of each item in each location. Compare this to the noun InventoryBalance, which represents system-maintained on-hand quantities.</p> <p>The purpose of the ReceiveInventoryCount is to transmit an inventory count to ERP/OMS from the actual physical inventory location. This count may be a cycle count or a physical count. This message may also apply to planned or unplanned inventory counts.</p>
Purchase Order	Send Acknowledge Change Get / Show	<p>The PurchaseOrder business object communicates an order to purchase goods from a buyer to a supplier. The PurchaseOrder carries information to and from the buyer and supplier. Once both Parties agree to the order's contents and specified terms and conditions, the Purchase Order is a legally binding document. The PurchaseOrder sends a customer's electronic form of a purchase order document to a supplier to purchase several Lines, each containing an Ordered Item.</p> <p>The purpose of the SendPurchaseOrder is to transmit a purchase order to a supplier's order management application.</p> <p>The purpose of the AcknowledgePurchaseOrder is to acknowledge receipt of the Purchase Order and reflect any changes. Commonly, the acknowledgment is generated by a third-party order management application and transmitted to a purchasing or procurement application.</p> <p>The purpose of the ChangePurchaseOrder is to request another business application to make changes to an existing Purchase Order. This change must refer to the original document and the item ordered. The change processing assumes the replacement of fields sent, except for the purchase order's identifying fields. If any of these key fields require changing (e.g., the PurchaseOrder Document ID or a LineNumber), that constitutes a cancellation of the request and the addition of another Purchase Order.</p> <p>The GetPurchaseOrder function enables a business application module to request information concerning a specific purchase order from another business application. The reply to this message is the ShowPurchaseOrder. Several environments may use this capability.</p>
ReceiveDelivery	Receive Send Acknowledge Get / Show	<p>The ReceiveDelivery business object represents a transaction for receiving goods or services. It may indicate receipt of goods in conjunction with a purchase order system. It provides general information about the ReceiveDelivery receipt document used to receive a specific quantity of material goods from a supplier. It also provides general information about the delivered item inventory, including the requested and actual quantities. This information usually directly references a line item or Delivery schedule line on a sales order, purchase order, or other inventory transfer document.</p> <p>The SendReceiveDelivery provides the ability to request an application/ system to create a ReceiveDelivery.</p> <p>The AcknowledgeReceiveDelivery may notify the shipping business partner that the shipment has been received by the customer or consignee destination and alert them to any discovered discrepancies. The acknowledgment may contain the full details of the receipt as created by the receiving party or just the discrepancies and other exception conditions. The AcknowledgeReceiveDelivery BOD supports receipt acknowledgments at the line item and ship unit levels. Intermediate transportation/logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents.</p> <p>The GetReceiveDelivery may request information about a specific expected (unreceived) or previously received goods delivery. The response to the GetReceiveDelivery request is the ShowReceiveDelivery. For expected deliveries, the ShowReceiveDelivery document content may be used as a receiving template or checklist to identify the quantity and shipping configuration of the expected goods. The ShowReceiveDelivery supports describing shipment content at either the line</p>

		<p>item level or the ship unit level. Intermediate transportation/logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents.</p>
WarehouseStockTransferShippingAdvice	Send	<p>The SendWarehouseStockTransferShippingAdvice business object is the equivalent of the EDI 943 Warehouse Stock Transfer Shipping Advice, which a brand, supplier, or manufacturer uses to notify third-party warehouse or third-party logistics firm that the supplier is making a transfer shipment to the warehouse. A manufacturer may also use the EDI 943 Warehouse Stock Transfer Shipping Advice to authorize a warehouse to accept a return from a retail customer.</p>
WarehouseStockTransferReceiptAdvice	Receive	<p>The ReceiveWarehouseStockTransferReceiptAdvice business object is equivalent to the EDI 944 Warehouse Stock Transfer Receipt Advice, an electronic version of the Warehouse Stock Transfer Receipt Advice document. The EDI 944 provides a retailer with information to correctly record and modify product inventory levels at a public warehouse. Besides being used to confirm the receipt of a transfer shipment, the EDI 944 reports product overages and shortages, whether the product is damaged, and the specific product status, such as the warehouse lot number or related production codes.</p>
SalesOrder	Receive Change Get / Show	<p>The SalesOrder is an order or customer order; it is a step beyond a PurchaseOrder in that the receiving entity also coordinates sales information about the Order along with the Order itself. SalesOrder is intended to be used when an order needs to be communicated between business applications, and the PurchaseOrder terms and conditions and quantities have been agreed to. This agreement may occur in an electronic or by other means.</p> <p>The ReceiveSalesOrder provides the ability to request an application/ system to create a SalesOrder.</p> <p>The purpose of the ChangeSalesOrder is to request that another business application component make changes to an existing Sales Order. Processing Note: This change must refer to the original document and item ordered. The change processing assumes replacing fields sent, except those uniquely identifying the document and its line, schedule, or subline. These include the DocumentId and the LineNumber for the Line, Subline, and Schedule. If any of the Field Identifiers above require changing, that constitutes a cancellation of the request and the addition of another Sales Order.</p> <p>The GetSalesOrder enables a business application module to request information concerning a specific sales order from another business application. The reply to this BOD is the ShowSalesOrder.</p>
FulfillmentOrder	Send Change Cancel	<p>The FulfillmentOrder business object represents a list of materials to be retrieved ("picked") from various locations in a warehouse to fill a production order, sales order, or shipping order. A picking list includes general identifying information (header information) and line item details. PickList may refer to only the header and detail information, depending on the verb.</p> <p>The SendPickList process transfers the details of an individual Picking List from EVP to the warehouse management system.</p> <p>The purpose of the ChangePickList is to change the details of an individual Picking List from EVP to the warehouse management system.</p> <p>The purpose of the CancelPickList is to cancel the details of an individual Picking List from an EVP to a warehouse management system.</p>
Shipment	Receive Get / Show Send	<p>A Shipment business object identifies and describes a specific collection of goods to be transported by a carrier and delivered to one or more business partner destinations. A Shipment document represents the extent and content of "transportation work" to be done by the carrier. For transportation efficiency, a shipment document typically consolidates deliveries to multiple destinations within a certain geographic region and may provide carrier routing instructions to each delivery stop. A Shipment can be used to represent:</p> <ul style="list-style-type: none"> Truckload to a single Ship To location Less Than a Truckload to a single Ship To location Multi-Stop TL or LTL to multiple Ship To locations Parcel Carrier shipment to a single Ship location <p>The ReceiveShipment provides the ability to request an application/ system to create a Shipment.</p>

		<p>The GetShipment aims to enable an application to request specific Shipment information from another business application module. The response to the GetShipment is the ShowShipment.</p> <p>The purpose of the SendShipment is to synchronize Shipment information with other applications.</p>
WarehouseShippingOrder	Receive	<p>The WarehouseShippingOrder business object is the equivalent of the EDI 940 Warehouse Shipping Order, which a supplier sends to authorize a third-party warehouse or third-party logistics firm (3PL) to make a shipment to a retailer, grocer or distributor or a secondary location of the supplier. The EDI 940 (x12 940 Warehouse Shipping Order) can either convey instructions to ship the same product to multiple locations or to ship specific cartons to a single location.</p>
WarehouseShippingAdvice	Receive	<p>The WarehouseShippingAdvice business object is the equivalent of the EDI 945 Warehouse Shipping Advice sent by a third-party warehouse or logistics firm to notify a supplier that a shipment has been made. The EDI 945 (x12 945 Warehouse Shipping Advice) transaction provides the information the supplier needs to reconcile the quantity shipped against the quantity ordered, create an invoice, and generate an 856 Advance Ship Notice (ASN) transaction. Depending on the supplier requirements, the warehouse may report the complete order as shipped or shipped with exceptions.</p>
Invoice	Receive Cancel Send	<p>The Invoice business object invoices a customer for goods/services provided.</p> <p>The purpose of the ReceiveInvoice is to transmit an invoice from a supplier to a customer or an online seller to a retailer in a DSV scenario. I indicated that the receiver of the Invoice is to process the Invoice for payment.</p> <p>The purpose of the CancelInvoice is to publish the need to cancel an Invoice or one or more of its elements to a business application or system.</p> <p>The purpose of the SendInvoice is to transmit an invoice from a supplier. This information is passed on to keep the customer updated on the number of Invoices they have.</p>
Confirm	n/a	<p>The CONFIRM is a specialized business object that performs the function of a Functional Acknowledgment that may be returned for a given message, depending upon the value set in the sender message confirmation field value (Never, OnError, Always). The confirmed response details will be one per business object instance in the original sender message. The CONFIRM is used when communicating business data across systems to indicate :</p> <ul style="list-style-type: none"> Positive Confirmation <ul style="list-style-type: none"> I received your message I read it (it's well-formed) I can process it (the data is valid) Exception Confirmation <ul style="list-style-type: none"> I received your message I can't read it (it's NOT well-formed) I can't process it (the data is invalid) <p>It can be used when processing business messages and validating:</p> <ul style="list-style-type: none"> Message/ Schema format Data List Validation Business Rules Validation <p>In summary, everything is OK or NOT with the message sent. It is not the ACKNOWLEDGE verb that</p> <ul style="list-style-type: none"> Occurs further downstream in the overall business processing Occurs after the business application processing of the relevant noun <p>The CONFIRM indicates integration between systems, and the ACKNOWLEDGE indicates the required business process of the message has occurred. Neither of these is related to the</p>

Other business objects (nouns) that can be communicated using EVP on an exception basis:

- BOM** (Bill Of Material) - Used to communicate the items and associated quantities that represent a kit sales listing
 - Typically set up using EVP Catalog Management screens or imported using the Catalog Source import feature.

- **CarrierRoute** - Describes a scheduled journey that a particular goods transportation service provider (freight carrier) is requested to perform on behalf of a shipper, customer, or another transportation coordinator. The journey may contain one or more stops. Many possible business applications in several environments may use this capability. A **transportation planning system typically generates a CarrierRoute** to coordinate and optimize the movement of many goods shipments to reduce overall freight cost, improve delivery schedules, and ensure that a carrier's equipment is utilized efficiently. Other business systems typically interested in a CarrierRoute document include supply chain execution, supply chain visibility, warehouse management, and transportation management systems.
- **CommericalInvoice** - A Commercial Invoice is not an Invoice document but a shipment document between a shipper and carrier. A Commercial Invoice document identifies and describes a specific collection of goods to be transported by a carrier and delivered to one or more business partner destinations. A Commercial Invoice document represents the extent and content of "transportation work" to be done by the carrier. For transportation efficiency, a Commercial Invoice document typically consolidates deliveries to multiple destinations within a certain geographic region and may provide carrier routing instructions to each delivery stop.
- **Field** - represents any user data element to be synchronized across databases or systems. The specific field name and value are specified in the relevant message.
- **PriceList** - Defines a list of items with their base price, price breaks, discounts, and qualifiers. For each item, price breaks can be defined, above which certain discounts or overriding prices might apply. Price breaks can be defined in volume or dollar amount. Price list qualifiers specify for which catalog, customer, and effective dates this price list applies.
- **ProductAvialbility** - represents information on the availability of a specified item at a specified inventory location for a specified date. Product availability is typically needed in the processing of customer sales orders. It is used in this context as the object of an inquiry function. The ItemQuantity indicates the number of Items requested, and the AvailableQuantity indicates the number of available items.
- **RequestForQuotations** (RFQ)—This document describes goods or services a supplier desires. It includes the terms of the purchase, delivery requirements, identification of goods or services ordered, and their quantities. The RFQ noun is used with the Quote noun to form a Business-to-Business negotiation dialogue concerning the goods or services specified.
- **Quote** - a document describing the prices of goods or services a vendor provides. The Quote includes the terms of the purchase, delivery proposals, identification of goods or services ordered, and their quantities. The Quote noun is used with the RFQ noun to form a Business-to-Business negotiation dialogue conc“. For transportation efficiency, a ShippersExportDeclaration document typically consolidates deliveries to multiple destinations within a certain geographic region and may provide carrier routing instructions to each delivery stop.
 - The manifesting engine will typically create and communicate this document to the carrier.
- **ShippersLetterOfInstruction** - A ShippersLetterOfInstruction document identifies and describes a specific collection of goods to be transported by a carrier and delivered to one or more business partner destinations. A ShippersLetterOfInstruction document represents the extent and content of "transportation work" to be done by the carrier. For transportation efficiency, a ShippersLetterOfInstruction document typically consolidates deliveries to multiple destinations within a certain geographic region and may provide carrier routing instructions to each delivery stop.
 - The manifesting engine will typically create and communicate this document to the carrier.

Typical Message Formats

These are exposed as needed via channel adaptors to predefined API, EDI, files, etc. As this can be virtually anything, it's a challenge, as discussed, to say this is our standard integration format. But to help understand what is out of the box, here are three methods we expose and utilize regularly:

- EDI see details documented [here](#)
- .CSV see details documented [HERE](#)
- API see details documented [here](#)

We must accommodate Many channels with predefined message formats; when these are high-usage channels (Amazon, eBay), we will implement processors at the code level. However, many channels may only have to integrate as needed for a single client. Therefore, we provide very flexible configuration mapping tools that allow the data in the message to be cross-referenced to the internal EVP channel model.

File-Based Configuration

File-Based Configuration	
File-based configuration supports: <ul style="list-style-type: none"> • Uploading data from many different source locations, in various formats, with different delimiters, and with or without header rows 	Can create output files formatted as needed based on taking keyword names and mapping them into specific columns in a file

- Ability to map keywords and channel attributes to columns within the data file
- File processing can be used for catalog creation, inventory, order creation, and shipment confirmation processing

NameEagles Tool Warehouse Catalog Load

ConfigurationStep NotesRun HistoryServices

ProcessorItemFileLoaderChannelEagles Tool WarehouseSettings

Check XMLFormat XMLShow All Properties

<Processor|
TestMode="false"
Quote="0x22"
Delimiter="0x2C"
ProperCaseTitle="false"
CreateNewItem="false"
InputFile="I8710DATAFEED**"
InputFilesPattern="true"
HaveTitles="true"
SkuColumn="Item Number"
IgnoreMissingSkuValues="false"
TitleColumns="{Description 1} {Description 2}"
MfrNameColumn="Vendor Name"
UpcColumn="UPC Code"
MatchOnMerchantSku="true"
MatchOnUpc="true"
MatchOnMfrPartNo="false"
AllowSupplierSkuChange="false"
AllowManufacturerChange="true"
RenameFiles="true"
DeleteRemoteFiles="true"
MfrPartNoColumn="Manufacture's Item"
ServiceName="Catalog File"
CreateMissingManufacturerCodes="true">
<AttributeMap>
<Map Source="Item Weight" Target="Weight" />
<Map Source="Width" Target="Width" />
<Map Source="Height" Target="Height" />
<Map Source="Length" Target="Length" />
<Map Source="Freight Policy" Target="FreightPolicy" />
<Map Source="Warranty" Target="Warranty" />
</AttributeMap>
<PriceMap>
<Map Source="Price" Target="Cost" />
</PriceMap>
</Processor>

ConfigurationStep NotesRun HistoryServices

ProcessorFlatFileOrderExportChannelEagles Tool WarehouseSettings

Check XMLFormat XMLShow All Properties

<Processor
Delimiter="0x2C"
Quote="0x00"
IncludeTitle="true"
FileNamePrefix="Auuvo_I8710_"
FileExtension="csv"
CreateOutputSubdirectories="true"
ServiceName="Send Orders">
<ColumnMap>
<Column Name="order-id" Data="PurchaseOrderNumber" />
<Column Name="sku" Data="SupplierSku" />
<Column Name="quantity-purchased" Data="Quantity" />
<Column Name="ship-service-level" Data="ShippingServiceLevel" />
<Column Name="recipient-name" Data="ShipToFullName" />
<Column Name="ship-address-1" Data="ShipToAddress1" />
<Column Name="ship-address-2" Data="ShipToAddress2" />
<Column Name="ship-address-3" Data="ShipToAddress3" />
<Column Name="ship-city" Data="ShipToCity" />
<Column Name="ship-state" Data="ShipToState" />
<Column Name="ship-postal-code" Data="ShipToPostalCode" />
<Column Name="ship-country" Data="ShipToCountry" />
<Column Name="ship-phone-number" Data="None" />
Value="" />

Or can upload data from files that have no column titles or use absolute positioning.

ConfigurationStep NotesRun HistoryServices

ProcessorFlatFileFulfillmentLoaderChannelEagles Tool WarehouseSettings

DeleteRemoteFiles="true"
ServiceName="Receive Tracking Files">
<ColumnMap>
<PurchaseOrderNumber Index="16" />
<TrackingNumber Index="22" />

EDI Configuration Framework

EDI Configuration Framework

The EDI Configuration framework supports:

- Trading partner parameters for the channel
- The EDI documents that the channel uses.
- Depending on the channel, these documents could be inbound or outbound
- The EDI documents that are acknowledged with a 997 document
- How the EDI document segments and attributes are cross-referenced to channel keywords

This mapping can support the creation of multiple business records within EVP based on a single EDI document.

In this example, an EDI 850 Store Level PO will create an Outbound Order for each Store and an Outbound Shipment for each unique DC associated with the various store orders.

Channel: Bradley Caldwell

Test Output

Test Parse

Merchant Qualifier/ID

22

PROMAC736044

Supplier Qualifier/ID

12

7174037511

Format Options

Terminator

--

Delimiter

*

Comp Separator

:

TX Number

3,079

Break Lines

☐

Transactions

File Name Template

Transaction Config

Custom Processors

<Ext>

<Proc> ID="300">

<Property Name="ShippingDate" Number="1" ValueType="DateTime" />

</Property>

<Property Name="Carrier" Number="3" />

</Property>

<Property Name="HarrisLocation" Number="3" />

</Property>

<Property ID="REP" QualifierNumber="1" QualityValue="10" LevelOverride="0" />

</Property>

<Property Name="TrackingNumber" Number="2" />

</Property>

Flexible Process and Deployment Models

Within a single channel, it is possible to leverage protocol and message format on a per touchpoint basis or even the same touchpoint based on need, i.e., use a .csv file via SFTP to process a sizeable daily inventory file but then call a JSON-based RESTFul API to lookup delta changes to inventory that have occurred during the last ‘n’ minutes. The overall integration model allows the scaling and distribution of processors combined with the ability to configure them as single or multi-threaded deployments to support scaling where various sales, suppliers, and integration channels process millions of events/messages per hour across our SaaS deployment model.

Where needed, touchpoints can be deployed to support highly scalable processing such as repricing notification from Amazon that, via multi-threading, can handle multiple concurrent requests processing at the rate of 200 per second or availability lookup requests across various fulfillment models (pick from stock or send to drop-ship fulfillment) across multiple fulfillment locations in the USA taking ~200 milliseconds.

Message	Seconds
Root	0.0000
Calculate Solutions	0.0000
Find Availability	0.0000
Line: 1) 1 EA of ACE-9463613	0.0000
Supplier Ace	0.0156
Supplier Etail	0.0156
Supplier Amazon Fulfillment	0.0625
Supplier Dynamics GP	0.0625
Supplier Emery Jensen Distribution	0.0781
Location Emery Jensen Distribution (Princeton IL) (90% Min: 4 Dropship: Submit) = 20	0.0781
Availability Mode: Actual	0.0781
Inventory cost: = 53.75 USD	0.0781
Direct - excluded; location mode = Off	0.0938
CrossDock - excluded; location mode = Off	0.0938
Dropship - 20 Actual; Shipping USD	0.0938
ThreePl - excluded; location mode = Off	0.0938
Location Emery Jensen Distribution (Sacramento CA) (90% Min: 4 Dropship: Submit) = 7	0.0938
Location Emery Jensen Distribution (Fredericksburg PA) (90% Min: 4 Dropship: Submit) = 72	0.0938
Location Emery Jensen Distribution (Colorado) (90% Min: 4 Dropship: Submit) = 11	0.1094
Location Emery Jensen Distribution (Gainesville GA) (90% Min: 4 Dropship: Submit) = 7	0.1094
Calculate Service Level on supplier Emery Jensen Distribution - found Etail (Standard) -> Supplier (Standard)	0.1094
Calculate Service Level on supplier Emery Jensen Distribution - found Etail (Standard) -> Supplier (Standard)	0.2188
Calculate Service Level on supplier Emery Jensen Distribution - found Etail (Standard) -> Supplier (Standard)	0.2188
Calculate Service Level on supplier Emery Jensen Distribution - found Etail (Standard) -> Supplier (Standard)	0.2188
Calculate Service Level on supplier Emery Jensen Distribution - found Etail (Standard) -> Supplier (Standard)	0.2188
Calculate Service Level on supplier Emery Jensen Distribution - found Etail (Standard) -> Supplier (Standard)	0.2188

The EVP application also supports the concept of various types of inventory status tracking:

- On-Hand Quantity** - This is tracked at a per fulfillment location level, i.e., 30 in the West Coast warehouse and 20 in the East Coast warehouse; if needed, a secondary location for each physical warehouse is set up to track unsellable inventory, received but pending putaway, damaged, etc.
- Sellable Quantity** – This is calculated by removing ‘Open Demand’ from ‘On-Hand’ Quantity, and any change in this value will trigger an inventory to all sales channels
- Open Demand**—An item in a fulfillment location can have open demand but is not yet allocated for order fulfillment processing. This allows pending orders that may still need to clear payment processing, fraud checks, or customization confirmation to be added to the system; the required items will have an open demand, which triggers the system to publish updated sellable inventory to all sales channels. Open Demand can expire based on a cancel order event or after ‘n’ minutes/hours/days.
- Allocated** – An item in a fulfillment location can also have an allocated qty, which represents the number of orders that have been sent to that location for fulfillment; if the allocated quantity matches the On-Hand Quantity, the location will no longer be considered available sellable quantity or fulfillment; which can trigger changes in pricing or shipping service levels, i.e., have sellable inventory in the west and east coast warehouse so will offer 3-day service to the majority of the USA; but if east coast warehouse goes to zero; will only offer five days as standard service level for customer east of the Mississippi, but continue to offer three days to other customers. The dynamic assignment of sales channel regional shipping policies publishes the current service level.

Using this ability to track inventory via fulfillment location, EVP can handle multiple order fulfillment scenarios:

- Assign the sales order to the 3P's available stock is all allocated, send the relevant supplier a drop-ship PO for the supplier's closest fulfillment location to the customer.
- If needed, certain items based on cost, weight, or the need to customize on-demand may only be offered via a drop-ship model, which can, therefore, have different shipping costs or order fulfillment times. For example, it takes three days to ship from the warehouse versus one day if handling from stock.

The integration with these suppliers can be via several interaction models based on the capabilities of the supplier and their internal systems:

- **Traditional model** – Purchase Order sent to their host system via EDI or some other agreed interface model
- **Hybrid model:** The Purchase Order is sent to their host system via EDI or some other agreed-upon interface model. However, as they do not support the required carrier or service level, the order is automatically manifested, and the shipping labels are emailed/FTP to the supplier.
- **Artisan Suppliers** – If the supplier has limited internal systems or cannot handle external integration, they can access EVP via a web browser to view their POs, create a packing slip, and manifest the order, which will send a shipping label to their printer.

The integration with the sales and fulfillment channels can also support the concept of order options at the order header and order line item level. These order options can be used to pass data as needed to support different levels of customizable orders, including:

- Gift message at the order header level
- Gift message at the order line level
- Configurable item values for each item, i.e., buy 3 of XYZ and engrave a different name on each item.

It can also support the concept of kits, which are sold as a single item on the sales channel but are made up of multiple stock items that will be ordered based on the KIT Bill of Material (BOM). In the remaining sections of this document, we will cover the various touchpoints we support on a channel basis. We can also expose an API to support interactions with the system.

The remainder of this document details the channel types supported within EVP and the specific adapters that can be configured as needed on a client EVP instance. It should be noted that a new adapter instance can be implemented using the generic channel type adaptor shell, which significantly reduces the effort needed for any client-specific integration needs.
