

# Migrating to STEPXML-based OIEP

**Important:** With the 2023.3 release, the JSON outbound integration will no longer be available. It is advised that the individual who maintains PDX integration configurations set up the new STEPXML outbound integration and disable the old JSON outbound integration before the 2023.3 release.

This integration approach, released with STEP version 10.1-mp4, moves the responsibility of converting multi-context STEPXML to PDX native JSON from the STEP application to the PDX application to improve the overall throughput of product records transferred from STEP to PDX.

## Prerequisites

The new integration from STEP to PDX is included as part of the productdatasyndication-integration component version 7.0.29 or higher; and can as such be patched onto the target STEP environment as a single component upgrade through the 'spot' tool. The new component version requires a STEP baseline of 10.1 MP4 or higher.

To install the required component and learn about how Stibo Systems Professional Services can assist with this migration, contact your Stibo Systems representative

## Migration Preparation

The STEPXML-based OIEP should be configured and tested by an admin or super user who has knowledge about the system utilization. After reading this topic, begin by setting up the standard STEPXML-based OIEP as instructed in the **STEPXML-based OIEP** topic in this documentation.

**Important:** The PDX Pre-production environment should be used for both functional and performance testing, whereas the PDX Production environment should only be the target of a STEP Production environment processing live data.

For customers migrating from the older JSON-based integration to PDX, it may be prudent to have both integrations running in parallel during the test phase in order to easily compare the results of the data transfer using the two methods. In such a scenario, the URL properties for both integrations should be included in the sharedconfig.properties file of the relevant non-production STEP environment used for testing:

```
PDS.Url=1=https://api.pdx-preprod.stibosystems.com
```

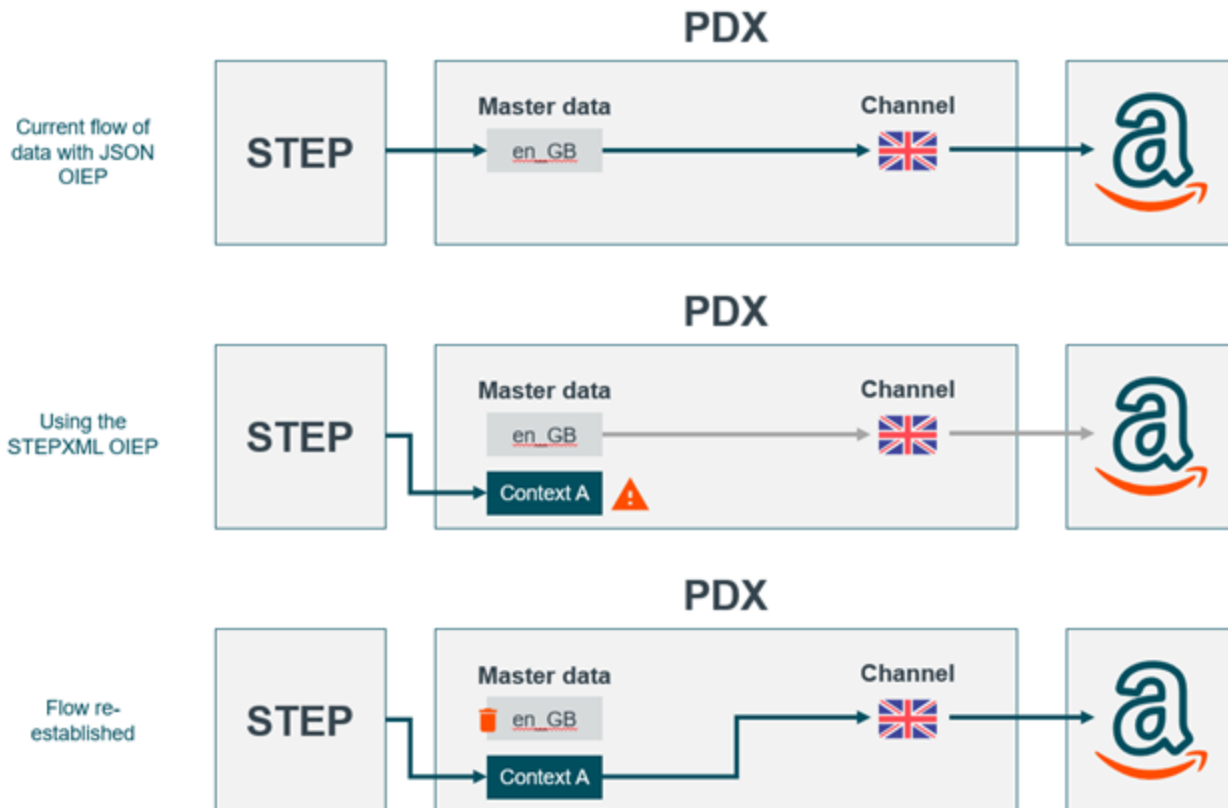
```
PDX.Url=1=https://api.pdx-preprod.stibosystems.com
```

The 'PDS' property determines the possible target environments of the JSON-based integration to PDX, while the 'PDX' property determines the possible target environments of the new STEPXML-based integration.

## Convert Locale-based to Context-based language layers

The STEPXML-based OIEP does not rely on locale-based mappings in PDX. With the STEPXML integration approach, new context-based language layers will be created in PDX.

**Note:** The newly created language layers need to be re-mapped on all channels. See below example diagram which illustrates how this flow needs to be converted.



The individual setting up the STEPXML-based integration can specify the language layer priority order in PDX. It is advised to set the new OIEP at the highest priority language layer(s) and complete the mappings in PDX. This will allow for updates to proceed on the re-established flow shown in the diagram above.

### Test:

**Important:** It is advised to perform this testing in the PDX Pre-production environment and not with a full product assortment.

Choose a subset of products to update and send through the STEPXML-based OIEP. Verify the data completeness and integrity in PDX master data. Then verify the data is as expected on a channel where the mappings to the context-based language layers have been completed. Expand the product selection pushed to PDX to the acceptance level to pass testing.

**Release:**

Once testing has concluded, complete the STEPXML-based OIEP configuration on the production environment. Ensure the context-based language layers are mapped in PDX Production. Verify connectivity and updates are flowing end to end as expected.

**Disable JSON-based OIEP:**

Allow the JSON-based OIEP to remain until a full product assortment update (whether overtime or in one larger update) has completed. Once confirmed, the old JSON-based OIEP should be disabled. Once the JSON-based OIEP has been disabled, the corresponding legacy language layers in PDX should be deleted. This makes the language mapping setup of future channels simpler, as there will be fewer options to pick from and removes the risk of mapping a legacy master data layer to a new channel layer.

# STEPXML-based OIEP

**Important:** With the 2023.3 release, the JSON outbound integration will no longer be available. It is advised that the individual who maintains PDX integration configurations set up the new STEPXML outbound integration and disable the old JSON outbound integration before the 2023.3 release.

The STEPXML-based OIEP produces cross-contextual STEPXML files that get enriched to make it self-contained. The final payload is then sent to PDX through the external API available, where it gets parsed into PDX-native JSON and imported.



The following topics describe how this integration is set up and configured to transfer the different types of product information that comprises a product record.

- Base Setup
- Handling Nested Data Structure
- Transferring Assets
- Transferring the Packaging Hierarchy
- Other OIEP Dependencies
- Validating the Configuration
- Known Limitations

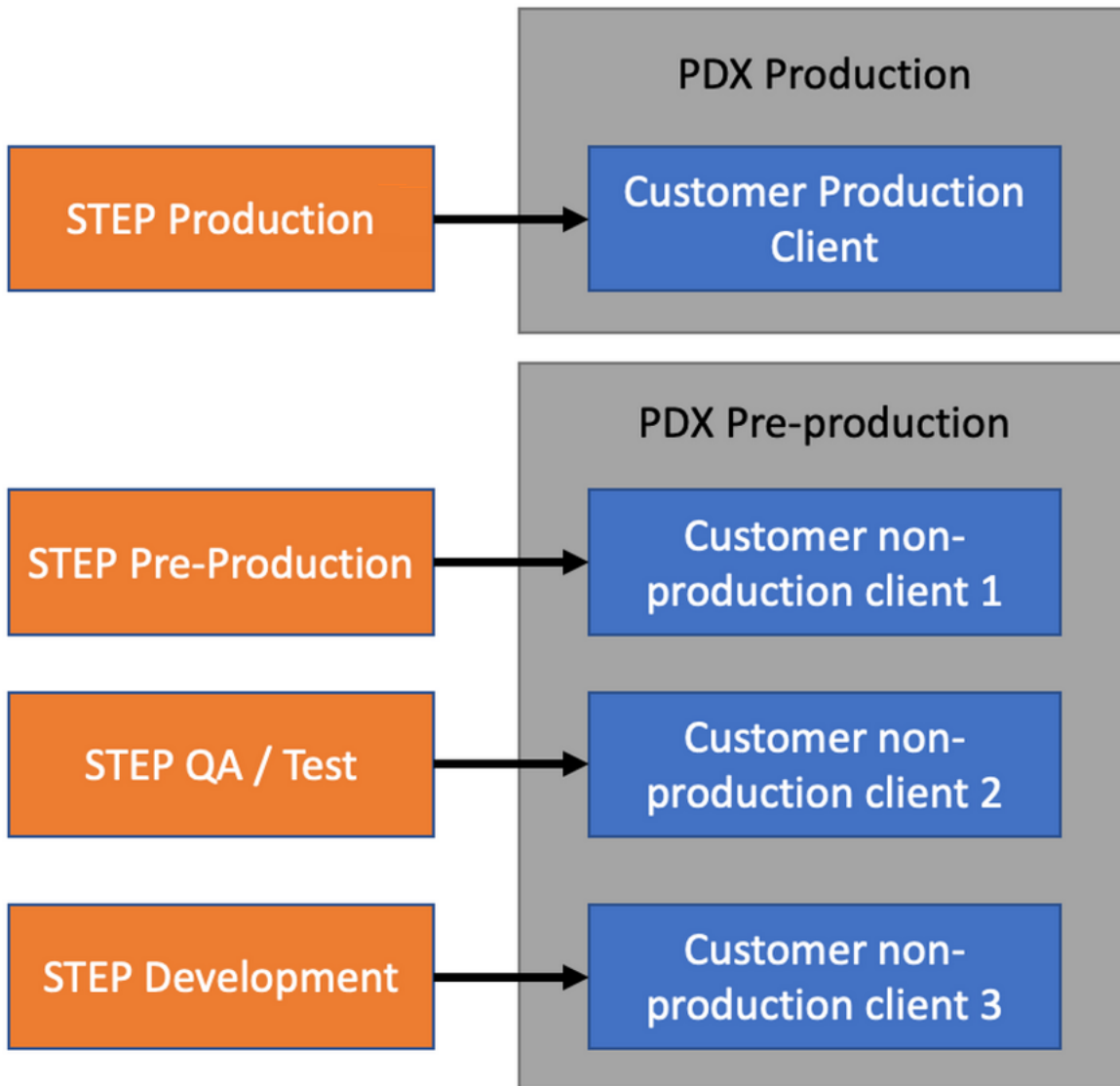
## Base Setup

The server-side setting of the `sharedconfig.property PDX.Url` determines the PDX environments eligible in the OIEP configuration of the delivery method. Two PDX environments are relevant to this setting:

- PDX Production (<https://api.pdx.stibosystems.com>)
- PDX Pre-production (<https://api.pdx-preprod.stibosystems.com>)

General setup will include one account / client on the Production environment, which processes and syndicates live data, while multiple non-production accounts / clients may exist on the Pre-production

environment. In the image below, those in orange represent STEP environments, those in Gray represent PDX environments, and those in blue represent PDX accounts.



The sharedconfig.property on the STEP Production environment would specify the PDX Production environment (containing the customer’s production account / client) as the valid target of the PDX OIEP configured, as seen in the example below:

```
PDX.Url=1=https://api.pdx.stibosystems.com
```

To ensure product information maintained in all the contexts selected for syndication in the PDX OIEP setup can be transferred, the following property should be set to ‘false’:

```
PDXDelivery2.LocaleChecking=false
```

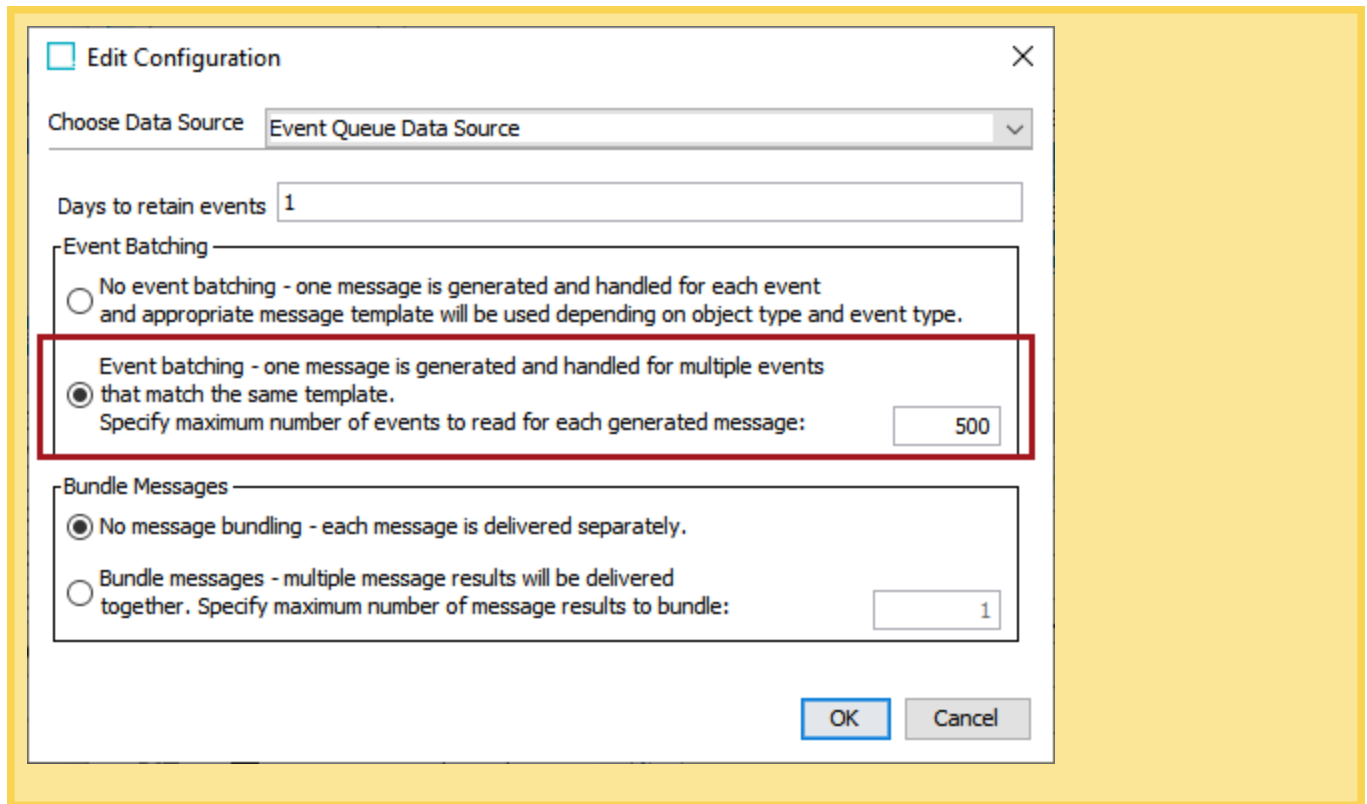
Once these are set, continue setup by following the steps below:

1. **The STEPXML Type:** Ensure that the OIEP always sends cross context formatted STEPXML to PDX by setting the 'Context Mode' to 'Cross Context Format' as highlighted below:

The screenshot shows the configuration interface for 'PDX OIEP for English'. The 'Configuration' tab is selected, and the 'Context Mode' is set to 'Cross Context Format'. The following table represents the configuration data shown in the interface:

Property	Value
Process Engine	STEP Exporter
Error reporter	Not Defined
Schedule	Not scheduled
Queue for endpoint	OutboundQueue
Queue for endpoint processes	Out
Transactional settings	Strict
Maximum number of threads	1
Maximum number of waiting processes	1
Maximum number of old processes	100
Maximum age of old processes	1w
<b>Context Mode</b>	<b>Cross Context Format</b>
Contexts	English
Workspace	Approved

**Important:** Ensure the event batching is set to a maximum of 500 to ensure optimal performance of the integration (see the screenshot below).



2. **The STEPXML Template:** The Format used in the PDX integration is Advanced STEPXML, and the template should always contain the following tags:

- ContextList
- CrossReferenceTypes
- UnitList
- AttributeList

The following is an example template:

```

1 <STEP-ProductInformation ResolveInlineRefs="true" FollowOverrideSubProducts="true">
2 <ContextList ExportSize="Minimum"/>
3 <CrossReferenceTypes ExportSize="Minimum"/>
4 <UnitList ExportSize="Minimum"/>
5 <AttributeList ExportSize="Minimum"/>
6 <Products ExportSize="Minimum" FlattenHierarchy="false">
7 <Product/>

```

```

8 </Products>
9 <Assets ExportSize="Referenced"/>
10 </STEP-ProductInformation>

```

3. **The Pre-processor:** The Pre-processor used should be the 'PDX Pre-Processor 2.'

It has a single setting called 'Only Changes.' If this is set to 'No,' then all products in the event queue will be sent. If it is set to 'Yes,' then only products with a changed revision will be sent.

4. **The Delivery Method:** When setting up the Delivery Method of the PDX OIEP, ensure that the 'Product Data Exchange 2' is chosen.

- Select the appropriate target PDX environment
- Enter the username and the API password of a PDX user associated with a relevant account on the targeted environment.



**Note:** If a HTTP Proxy has been configured, you're able to choose it in the "Proxy Config" dropdown list. For more information on how to configure a proxy, please see the HTTP Proxy Configurations section.

- Specify which of the contexts included in the OIEP configuration is the 'default context.' The data in this context will be used to populate the default language layer in PDX.

**Important:** Channel assignment rules in PDX will be evaluated based on the values available in the default language layer.

- Set the 'Upload only changed assets' setting to 'Yes,' if you only want to include referenced asset objects where either attribute values or the asset content has changed.

## Handling Nested Data Structures

Nested data structures are sometimes required to capture pieces of product information. A good example is nutritional information describing the contents of a product at various scales. Consider the GDSN modeling of this data:

```
Nutritional Information: group{
  Preparation State: lov,
  Daily Value Intake Reference: string+,
  Nutrient Basis Quantity Type Code: lov,
  Nutrient Basis Quantity: decimal,
  Serving Size: decimal[],
  Serving Size Description: string+,
  Nutrient Detail: group{
    Nutrient Type Code*: string,
    Percentage of Daily Value Intake: decimal,
    Measurement Precision: lov,
    Nutrient Quantity Contained: decimal[],
    Daily Value Intake Percent Measurement Precision Code: lov,
    Nutrient Value Derivation Code: lov,
    Description On Nutrient Qualifier: string[],
```

Nutrient Source: string+

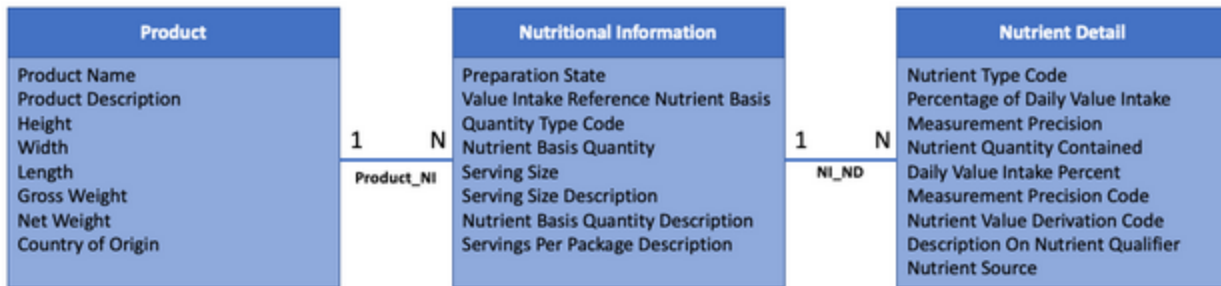
} [],

Nutrient Basis Quantity Description: string+,

Servings Per Package Description: string+

} []

To model this data as accurately as possible, the STEP data model is currently required to have a referential nature (i.e. using references to other objects) as illustrated below:



The screenshot shows a software interface with a tree view on the left and a 'References' table on the right. The tree view shows a hierarchy of products, with 'Cinnamon Sprinkles Cereal' selected. The 'References' table is titled 'Cinnamon Sprinkles Cereal rev.0.10 - References' and has tabs for Product, Data Containers, References, Referenced By, Images & Documents, Commercial, and Tables. The 'References' tab is active, showing a table with columns for Reference Type, Target, and other fields. A red box highlights a row in the 'Ungrouped Product References' section, showing a reference to 'Nutritional Information' with a target of 'Nutritional Panel 1'.

The screenshot displays the STIBO SYSTEMS software interface. On the left is a 'Tree' view showing a 'Primary Product Hierarchy' with categories like Apparel, Audio Visual Equipment, Bundle Hierarchy, Foodservice, Ready To Eat, and Nutritional Data. Under 'Ready To Eat', there is 'ACME Cereal' and 'Cinnamon Sprinkles Cereal'. Under 'Nutritional Data', there are 'Nutrient 1' through 'Nutrient 6' and 'Nutritional Panel 1' through 'Nutritional Panel 3'. A red arrow points from 'Nutritional Panel 1' in the tree to the right-hand panel.

The right-hand panel is titled 'Nutritional Panel 1 rev.0.4 - Product'. It has several tabs: 'Product', 'References', 'Referenced By', 'Images & Documents', and 'Commercial'. Under the 'References' tab, there are sub-tabs for 'Tables', 'Proof View', 'Status', 'State Log', and 'Tasks'. The main content area shows 'Ungrouped Product References' with a table:

Reference Type	Target	
	Nutrient 1	X
	Nutrient 2	X
> Nutrient Details +	Nutrient 3	X
	Nutrient 4	X
	Nutrient 5	X
	Nutrient 6	X

Below this table are sections for 'Index Words' (with 'Inherited From' and 'Index' fields), 'Publications' (with 'ID' and 'Name' fields), and 'Linked Attributes from Product Hierarchy' (with 'Display Sequence', 'ID', 'Name', and 'Attribute Groups' fields). There are also links for 'Add Index Word', 'Link to Publication or Section', and 'Link to Attribute'. At the bottom, there is a section for 'Linked Attributes from Classification Hierarchy'.

At the bottom left of the screenshot, the status 'Ready' is displayed.

The referential structure needs to be included in the Advanced STEPXML Template of the PDX OIEP, by embedding the referenced data. Consider the example below:

```

1 <STEP-ProductInformation ResolveInlineRefs="true" FollowOverrideSubProducts="true">
2 <ContextList ExportSize="Minimum"/>
3 <CrossReferenceTypes ExportSize="Minimum"/>
4 <UnitList ExportSize="Minimum"/>
5 <AttributeList ExportSize="Minimum"/>
6 <Products ExportSize="Minimum" FlattenHierarchy="false">
7 <Product>
8 <Name/>
9 <Values/>
10 <ProductCrossReference Type="NutritionalInformationReference" ExportSize="Minimum">
11 <MetaData/>
12 <Product Referenced="True" Embedded="True">
13 <Values/>
14 <ProductCrossReference Type="NutrientDetailsReference" ExportSize="Minimum">
15 <MetaData/>
16 <Product Referenced="True" Embedded="True">

```

```

17 <Values/>
18 </Product>
19 </ProductCrossReference>
20 </Product>
21 </ProductCrossReference>
22 </Product>
23 </Products>
24 <Assets ExportSize="Referenced"/>
25 </STEP-ProductInformation>
    
```

**Important:** In the examples used, the referential structure is modeled using STEP Objects of the Product type tied together by Product cross-references, but Entity-based structures would be equally valid for transferring a nested data structure to PDX.

On the PDX side, the attribute values of each reference and its target object are consolidated to form a row of values in a Composite Attribute. PDX Composite attributes closely resembles multi-valued Data Containers in STEP, but they are able to contain other Composite Attributes, thus forming a 'tree' of data.

The image shows three sequential screenshots of a web application interface, illustrating a drill-down process from a list of products to detailed nutritional information.

**Top Screenshot:** A 'Master data' list view. A table with columns 'ID', 'NAME', and 'NUTRITIONAL INFORMATION' is shown. The first row is highlighted with a red box: ID: SalesItem-105807, NAME: Cinnamon Sprinkles Cereal.

**Middle Screenshot:** A detailed view of 'Cinnamon Sprinkles Cereal' (ID: SalesItem-105807). The path is 'Syndication / Nutritional Information'. A table with columns 'ANALYTICAL\_AND\_C...', 'DAILY INTAKE REFERENCE', 'SERVING SIZE', 'SERVINGS PER PACKAGE DESCRIPTION', 'SERVING SIZE DESCRIPTION', and 'PREPARATION STATE' is shown. The first row is highlighted with a red box: ANALYTICAL\_AND\_C...: 2, DAILY INTAKE REFERENCE: 22 Gram, SERVINGS PER PACKAGE DESCRIPTION: 4, SERVING SIZE DESCRIPTION: Per Serving, PREPARATION STATE: Unprepared.

**Bottom Screenshot:** A further detailed view of 'Cinnamon Sprinkles Cereal' (ID: SalesItem-105807). The path is 'Syndication / Nutritional Information / Nutrient Details'. A table with columns 'NUTRIENT SOURCE', 'LABEL', 'NUTRIENT TYPE CODE', 'NUTRIENT QUANTITY CONTAINED', 'DAILY VALUE INTAKE PERCENT', and 'MEASUREMENT PRECISION CODE' is shown. The first row is highlighted with a red box: ANALYTICAL\_AND\_CAL...: 1, LABEL: ENER-, NUTRIENT QUANTITY CONTAINED: 13, DAILY VALUE INTAKE PERCENT: 2, MEASUREMENT PRECISION CODE: Approximately.

## Managing Composite Attribute Names Shown in PDX

By default, composite attributes are titled according to the name of the given STEP reference type, or the ID of the reference type if no name is available. However, the recommendation is to create a meta data attribute for the reference types tying together the nested data structures that explicitly sets the ID of the Composite Attribute in PDX. This can be done through the following steps:

1. **Creating the metadata attribute:** A new metadata attribute is needed to store the desired name of the composite attributes to be shown in PDX. Find an appropriate attribute group and create a new attribute within it. It should be a single-valued description attribute with a suitable name and no dimension dependencies, that can be used to store text.
2. **Adding Reference Type Validity:** Go to the Object Types & Structures section in the System Setup tab, and expand the Basic Object Types subsection. Find the 'Reference-Type' object type, and add the newly created metadata attribute to the list of valid attributes.
3. **Adding Metadata Values to the Reference Types:** The reference types constructing the nested data structure are then updated with the appropriate names for the composite attributes they will be converted into on the PDX side, by entering a value for the newly created metadata attribute.

For example, two references used to model nutritional information, i.e. 'Product to Nutritional Information' (NutritionalInformationReference) and 'Nutritional Information to Nutrient Details' (NutrientDetailsReference) will be given the values 'Nutritional Information' and 'Nutrient Details' respectively.

4. **Adding to the Sharedconfig.property file:** In order to inform PDX of which attribute should be used to determine the composite attribute name, the property 'PDSDelivery.CompositeAttributeID' must be added to the sharedconfig.properties file. This will have to be set to the STEP attribute ID of the newly created metadata attribute.

In this example case, the addition to the sharedconfig.property file would be the following:

```
PDSDelivery.CompositeAttributeID=PDXCompositeAttribute
```

5. **Validate the Composite Attribute Names:** Lastly, a product with the nested data structures can be pushed to PDX through the OIEP, allowing validation of the composite attribute names (the column headers) which reflect the values added to the reference types in Step 3.

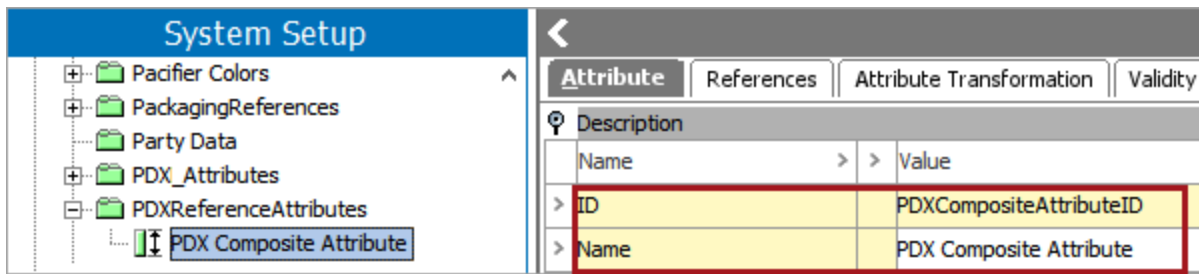
Specifically, a description attribute must be created, made valid for the reference types of interest, and the ID of this attribute must be added to the **PDSDelivery.CompositeAttributeID=[attribute\_id]** property in the sharedconfig.properties file on your STEP application server. For example:

```
PDSDelivery.CompositeAttributeID=PDXCompositeAttribute
```

The 'PDX Composite Attribute' attribute can be given any ID or name and can be created anywhere on your system.

**Example 1:** The name of the attribute is 'PDX Composite Attribute,' the ID of the attribute is 'PDXCompositeAttribute,' and it has been created within an attribute group named 'PDX Reference Attributes.'

A value can be set for the created meta data attribute, allowing the user to change the composite attribute ID in PDX.



**Example 2:** The 'PDX Composite Attribute' attribute value is set, which will now be included as a description attribute within the 'NutritionalInformationReference' and 'NutrientDetailsReference' reference types, changing the IDs of the two composite attributes in PDX to 'Nutritional Information' and 'Nutrient Details' respectively.

**Note:** The 'PDX composite attribute ID' attribute is made valid on the reference type *object*, not on the reference type *link* between the source and target product.

Once the bulleted items listed above are implemented, create a value for the 'PDX Composite Attribute' attribute, which will now be included as a description attribute within the 'Product to Product Case' reference type. In the example below, the user has assigned the value 'ProductCase' to the 'PDX Composite Attribute.'

**System Setup**

- User
- User Group
- Value Generator Configuration Type
- Web Service Endpoint
- WebUI Steppers Configuration Type
- Workspace
- XSLT Stylesheet
- Commercial Types
- Entity user-type root
- Event Queue Object Types
- Primary Product Classification
- Product-Overrides
- Publication group types
- Publication section types
- Publication types
- Setup Group type root
- Tags
- Units
- Users & Groups
- Reference Types
  - Product Reference Types
    - BicycleWheel
    - Case To Child
    - DC\_Reference\_Type
    - Package To Material
    - Pallet To Child
    - Product To Package
    - Product to Product Case**
    - Product to Product Power

**Reference Type** | Validity | Log

**Description**

Name	Value
ID	Product to Product Case
Name	Product to Product Case
Last edited by	2020-11-04 13:48:42.047 by USERK
Externally Maintained	No
Dimension Dependencies	
Allow multiple references	Yes
Mandatory	No
Inheritance	None
Attribute Completeness Score	123
Completeness Score	123
PDX Composite Attribute	abc ProductCase

**In Attribute Groups**

ID	Name
<a href="#">Add Attribute Group</a>	

**Valid Attributes**



ID
<a href="#">CaseSupplier</a>
<a href="#">Add Attribute</a>

With the 'PDX Composite Attribute' now assigned a value, when the 'Bose Soundlink Micro' (Black) product is exported to PDX, the composite attribute on the 'Product to Product Case' reference will be titled 'ProductCase,' based on the value of the 'PDX Composite Attribute' attribute.

✕ Bose Soundlink Micro (Black)

Summary
Product attributes
Packaging
Product variants
Digital assets

Ungrouped attributes

<b>ProductCase *</b>	<u>AenIlosi SKB Gator Penguin</u> 
<b>ProductImage *</b>	<u>BTSImage2, BTCImage1</u> 

For more information on the PDX Status Attribute Group attributes, refer to the **PDX Channel Status Monitoring** section of this documentation. For more information on creating attributes, refer to the **Creating Attributes** topic in the **System Setup** documentation.

## Transferring Assets

Assets can easily be included in the outbound data to PDX by adding the appropriate tags to the Advanced STEPXML template of the PDX OIEP. For more information, refer to **Advanced STEPXML Format** topic in the Data Format documentation.

The integration to PDX also allows transfer of context-specific assets, when the dimension dependency is managed on the reference type, refer to the **Dimension Dependency on a Reference Type** section of the **Dimension Dependent Reference and Link Types** topic for additional details.

If the assets are stored in an external DAM integrating with STEP, the following property can be added to the sharedconfig.properties file on the STEP application server, in order to disable the default behavior of STEP pushing asset binaries to PDX:

```
PDSDelivery.IncludeAssetContentsFromDAM=false
```

This is the preferred option in cases where an asset URL is stored on the relevant asset objects allowing PDX to automatically download the assets from the DAM. If this URL can be forwarded to the recipients downstream of PDX, it would significantly reduce the total amount of data transferred between the parties involved.

## Transferring the Packaging Hierarchy

In order to transfer the packaging hierarchy associated with the product information, the Packaging Component Model has to be configured with the appropriate object types, reference types, and quantity metadata attributes valid for those object types. For more information on configuring the Packaging Component Model, refer to the **Configuring the Packaging Component in STEP Workbench** topic in the **Web User Interface** documentation.



Packaging - Component Model Configuration			
Component Model Configuration			
Name		Value	Description
Packaging object types		Case	List of product object types that are used in the Packaging Component Model
		Display Shipper	
		Each	
		Inner Pack	
		Pallet	
Quantity of the next lower level package		Package Reference Quantity	Description attribute containing the quantity of the next lower level packages
Packaging reference types		Pallet to Child	Reference types that relates a packaging object to its child packaging object
		InnerPack to Child	
		Case to Child	
<a href="#">Edit</a>			

**Note:** The lowest level of the syndicated packaging hierarchy should be the 'Each' in order to avoid products being ineligible for submission to recipients downstream of PDX.

The relevant object types can then be added to the list of Object Event Types of the PDX OIEP, so updates to packages trigger the export of the change to PDX.

Example:

Conditions for output template

All object types

Specify

Browse Search

Each (Each) Search

Advanced

Name

> **Each ID = Each**

Case

- Each
- Inner Pack
- Mixed Module
- Pallet
- Shrink Wrap

Event Types

Create  Test

Modify

Delete

OK Cancel

The appropriate packaging references has to be added to the Advanced STEPXML Template of the PDX OIEP. Consider the below example:

```

1 <STEP-ProductInformation ResolveInlineRefs="true" FollowOverrideSubProducts="true">
2 <ContextList ExportSize="Minimum"/>
3 <CrossReferenceTypes ExportSize="Minimum"/>
4 <UnitList ExportSize="Minimum"/>
5 <AttributeList ExportSize="Minimum"/>
6 <Products ExportSize="Minimum" FlattenHierarchy="false">
7 <Product>
8 <Values/>
9 <ProductCrossReference Type="Pallet_to_Child"/>
10 <ProductCrossReference Type="InnerPack_to_Child"/>
11 <ProductCrossReference Type="Case_to_Child"/>
12 </Product>
13 </Products>

```

```
14 | <Assets ExportSize="Referenced"/>  
15 | </STEP-ProductInformation>
```

## Other OIEP Dependencies

The PDX OIEP automatically includes information of the Primary Product Image and the Attribute Help text retrieved from the baseline STEP configuration:

System Settings

System Settings Log

Product Information Manager Default Settings

Name	Value
> Enforce Mandatory Check for Attributes, References and Links	none
> Product Editor, Group attributes by top group	N
> Localize numbers with thousand delimiter when localizing exports	Y
> Localize dates when localizing exports	Y
> Report Save As CSV Character Set	client-locale
> Default Attribute to use as Display Sequence Attribute	DisplaySequence
> Default Completeness Metric	Default Completeness
> Conditional Validity Attribute	ConditionAttribute
> Block Attribute Groups with more than 1000 attributes	Y
> Use full pathname for classes on Product References Tab	Y
> Pass through unconverted Special Characters and Tags (Y) or discard th...	N
> Product Attribute Help metadata attribute	Attribute Help Text (AttributeHelpText)
> Attribute indicating the Priority of Product Variant Attributes	
> Show both name and ID in the PIM navigator	N
> Comma separated list of node types for which to show both name and ID	
> Attribute Header Column Width	250
> Reference Type Column Width	120

Translation Settings

GDSN Settings

Terms List Settings

Web Services

Web UI Settings

Primary Image Type

Name	Value
> Primary Image ReferenceType	ProductImage

The integration to PDX will transfer available help text stored in the meta data attribute selected, in the example shown above, it is 'Attribute Help Text,' for the product attributes included in the export. However, only the attribute help text in the Default Context will be imported to PDX. This will make the help text available when viewing the PDX Master Data. For more information, refer to the **Base Setup** topic in this documentation.

In the example below:

1. An attribute is shown with a meta data attribute storing help text.

The screenshot shows the 'Country of Origin - Attribute' configuration window. The 'Attribute Help Text' field is highlighted with a red box, containing the text: 'A description of the geographic area the item may have originated from or has been processed.'

The help text is then visible on the column headers in PDX Master data as seen in the two locations below:

The screenshot shows the PDX Master data interface. The 'COUNTRY OF ORIGIN' column header is highlighted with a red box, showing the help text: 'A description of the geographic area the item may have originated from or has been processed.'

## Mapping to Nile.com

Create mappings from master data columns to Nile.com

Nile.com columns
Master data
count
Details

Nile.com columns	Master data	count	Details
Product Title	G	Suggested	<p>Channel data</p> <p><a href="#">nile.com</a></p> <p>No description available</p> <p>Master data</p> <p><b>Country of Origin</b> (ID: Country of Origin)</p> <div style="border: 1px solid red; padding: 5px;"> <p>A description of the geographic area the item may have originated from or has been processed.</p> </div> <p>Transformation results</p> <p>Germany In</p>
Production year	G	Other	
Price	G	Country of Origin	
Sellable in store	G		
Sellable online	G		
Front	G		

### Your mappings

Column	Type	Mapping	Action
Size	G	Size	X
Sellable from	G	Available From	X
Battery Full Name	G	Battery	X
Rounded Height (cm)	G	Height	X
Depth (in)	G	Depth	X

Cancel Done

- Similarly, the designated primary image reference type, 'ProductImage,' will determine what images shows up in the thumbnails of the PDX UI.

In the screenshot below, there is a product with multiple asset references.

The screenshot shows a 'Tree' view on the left with a hierarchy: Primary Product Hierarchy > Products > Apparel > Audio Visual Equipment > Portable Audio/Video > Televisions > SU4-LCD > SU4-P7000 > SU4-P7000-55. The main window is titled 'SU4-P7000-55 rev.0.9 - References'. It contains several tabs: Product, Data Containers, References, Referenced By, Images & Documents, Commercial, Tables, Proof View, Status, State Log, and Tasks. The 'References' tab is active, showing three sections: 'Images References', 'Product References', and 'Product To Classification Link'. The 'Images References' section has columns: Reference Type, Target, and Photographer. It lists 'Primary Product Image' (targeting 'P\_P7000') and 'Product Image' (targeting 'P\_P7000 es' and 'P\_P7000 fr'). The 'Product References' section has columns: Reference Type, Target, and Quantity. It lists 'Accessories' (targeting 'AE Bracket - max60' and 'AE Bracket - max80') and 'Sell Side - Buy Side' (targeting 'Series P7000 model 55'). The 'Product To Classification Link' section has columns: Reference Type, Target, and Front Page. It lists 'Web Link' (targeting an image icon).

When viewed in the master data grid view in PDX, a user see the asset references under the corresponding columns.

The screenshot shows the 'Master data' interface. At the top, there is a search filter 'Starts with SalesItem-101326' and a language dropdown set to 'EN English'. Below this is a table with the following structure:

ID	PRODUCTIMAGE *	PRIMARYPRODUCTIMAGE *
SalesItem-101326	100512, 100515, 100514	100516

Red boxes in the original image highlight the 'PRODUCTIMAGE' and 'PRIMARYPRODUCTIMAGE' columns and their corresponding data rows.

If the Primary Image Reference Type is not defined in STEP, assets will still be transferred and available to recipients downstream of PDX.

## Validating the Configuration

Before enabling the configured PDX OIEP, it might be worth validating the configuration and settings required for the integration to be successful. Please consider the topics below.

### STEP Application Properties

The sharedconfig.properties should contain

- Mandatory properties
- The declaration of the STEP attribute controlling the PDX composite attribute naming
- The setting to exclude the transfer of DAM-hosted binaries

Example:

```

=====
# PDX Settings
=====

PDX.Url=https://api.pdx-preprod.stibosystems.com

PDXDelivery2.LocaleChecking=false

PDSDelivery.CompositeAttributeID=PDXCompositeAttribute

PDSDelivery.IncludeAssetContentsFromDAM=false

```

### OIEP Advanced STEPXML Template

The Advanced STEPXML Template of the PDX OIEP should contain the

- Mandatory tags
- Embedded reference targets for nested data structures
- References to relevant assets
- References forming the packaging hierarchy

Example:

```

1 <STEP-ProductInformation ResolveInlineRefs="true" FollowOverrideSubProducts="true">
2   <ContextList ExportSize="Minimum"/>
3   <CrossReferenceTypes ExportSize="Minimum"/>
4   <UnitList ExportSize="Minimum"/>
5   <AttributeList ExportSize="Minimum"/>
6   <Products ExportSize="Minimum" FlattenHierarchy="false">
7     <Product>
8       <Name/>
9       <Values/>
10      <ProductCrossReference Type="Product_NI" ExportSize="Minimum">
11        <MetaData/>
12        <Product Referenced="True" Embedded="True">
13          <Values/>
14          <ProductCrossReference Type="NI_ND" ExportSize="Minimum">
15            <MetaData/>
16            <Product Referenced="True" Embedded="True">

```



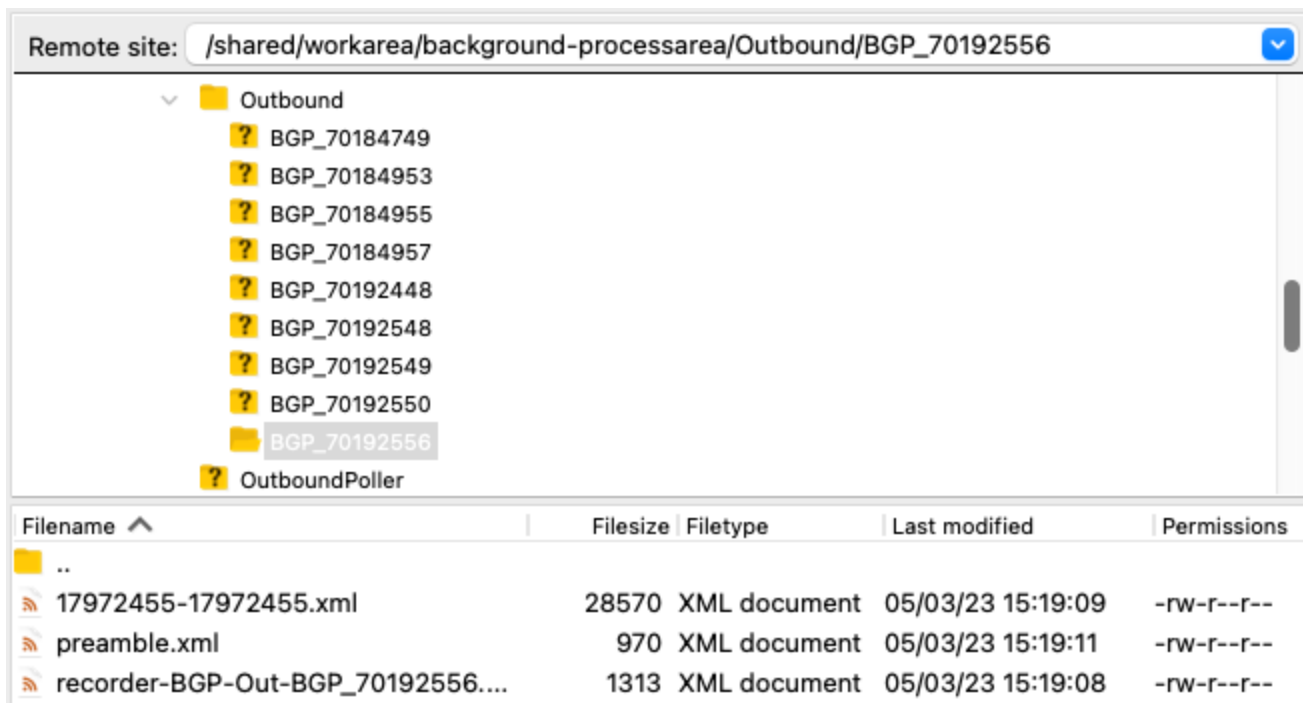
```

17         <Values/>
18     </Product>
19 </ProductCrossReference>
20 </Product>
21 </ProductCrossReference>
22 <ProductCrossReference Type="Pallet_to_Child"/>
23 <ProductCrossReference Type="InnerPack_to_Child"/>
24 <ProductCrossReference Type="Case_to_Child"/>
25 <AssetCrossReference Type="ProductImage"/>
26 </Product>
27 </Products>
28 <Assets ExportSize="Referenced"/>
29 </STEP-ProductInformation>

```

### The Preamble

In order for PDX to correctly interpret and parse the inbound STEPXML, STEP needs to enrich the OIEP-generated payload with critical information about the data model implemented. This enrichment is referred to as the 'preamble' and is stored in a separate file in the BGP folder.



The content of the preamble is a mix of data model and component configurations, as well as server-side settings.

Example:

```

1 <AuxiliaryTypes>
2   <PackagingTypes QuantityAttribute="Quantity">
3     <ReferenceType ID="Pallet_to_Child"/>
4     <ReferenceType ID="Case_to_Child"/>
5     <ReferenceType ID="InnerPack_to_Child"/>
6     <UserType ID="PA"/>
7     <UserType ID="CA"/>
8     <UserType ID="EA"/>
9   </PackagingTypes>
10 </AuxiliaryTypes>
11 <ConfigurationProperties>
12   <AttributeHelpAttributeID>AttributeHelpText</AttributeHelpAttributeID>
13   <DefaultContext>EN_GB</DefaultContext>
14   <PDSDelivery.AssetUploadedAttributeID>PDS Syndication
15   Time</PDSDelivery.AssetUploadedAttributeID>
16   <PDSDelivery.CategoryAttributeID>PDSCategory</PDSDelivery.CategoryAttributeID>
17   <PDSDelivery.CompositeAttributeID>PDXCompositeAttribute</PDSDelivery.CompositeAttributeID>
18   <PDS_Status_Attribute_Group>PDS Status Attribute Group</PDS_Status_Attribute_Group>
19   <PrimaryImageType>ProductImage</PrimaryImageType>
20 </ConfigurationProperties>

```

**Important:** It's recommended to review the preamble of initial exports to PDX to ensure that all elements are present / populated, as missing declarations often results in incomplete data transfers.

## Known Limitations

Below is a list of known limitations for the PDX STEPXML based outbound integration endpoint.

### Dimension dependent Asset content

It is not currently possible to transfer context-specific images by making the asset content dimension dependent as described in the **Asset Dimension Dependencies** subsection of the **Maintaining Assets** topic in the **Getting Started** documentation. However, a similar result can be achieved by making the asset reference type dimension dependent instead.

### Transfer of Product Families

It is not currently possible to transfer objects modeling a product family to PDX. However, if the ID of the family object is included on the object representing the core product record, PDX will allow a virtual grouping of products based on the value of such an attribute.

### **LOV Value IDs**

It is not currently possible to transfer the value IDs of attributes using the LOV Base Type validation. Only the raw value of the attribute is included in the exports to PDX.

### **Product-to-Product References**

It is not currently possible to transfer information about a products relationship to other products (e.g. accessories, bundles it's included in, cross-sell opportunities, etc.) modeled through references in STEP with identical source and target object types.

### **Data containers**

It is not currently possible to transfer data stored in STEP Data Containers. This means that the data either needs to be modeled as flat attributes on the Object itself, or as a referential structure. Refer to the **Handling Nested Data Structures** topic in this documentation for more information.

### **Dimension dependent STEP Names**

It is currently not possible to transfer language-specific attribute, reference, and unit names to PDX. The STEP Names available in the 'default context' are the only source of information used in the data transfer. For more information, refer to the Base Setup topic in this documentation.

### **Classification references**

It is currently not possible to transfer references from products to classifications. Product-relevant information stored in the classification structure, should be made available on the Object representing the core product record through Business Actions or Calculated Attributes.