Table of Contents

XML Path Language (XPath) is an expression language that can address parts of an XML document. The purpose of an XPath expression is to pull a value from the appropriate field in the EFI Pace Object Model so the value can be used in a quantity calculation.

Refer to the following topics to learn more about the EFI Pace Object Model and XPath expressions:

Table of Contents.................................................................................................................. 3
EFI Pace Object Model Basics.................................................................................................. 5
  Fields and Container Objects............................................................................................... 5
  Ancestor Objects .................................................................................................................. 5
  Related Objects .................................................................................................................... 7
  Base Objects ......................................................................................................................... 8
  Data Types ............................................................................................................................ 8
Diagram: Generic Object Model Tree......................................................................................... 9
EFI Pace Object Model Browser ............................................................................................... 10
Steps to Creating an XPath Expression .................................................................................. 11
  Determine what the XPath expression needs to do............................................................. 11
  Determine field names .......................................................................................................... 12
  Determine the data type of the result field .......................................................................... 13
  Determine the relationship of the fields to the base object ............................................... 13
  Write the expression in simple terms ............................................................................... 14
  Write the expression in XPath ............................................................................................ 14
  Test the XPath expression .................................................................................................. 14
  Enter the XPath expression in EFI Pace .............................................................................. 15
  Use an XPath expression ..................................................................................................... 17
XPath expression advanced formatting .................................................................................. 18
  Relational Operators ......................................................................................................... 18
  Arithmetic Operators ......................................................................................................... 18
  Rounding functions ............................................................................................................ 18
XPath Expressions for Estimating .......................................................................................... 20
  Estimating Object Tree ....................................................................................................... 20
  Prepress Operations ............................................................................................................ 20
  Finishing Operations .......................................................................................................... 20
  Price List Items ................................................................................................................... 20
  Estimating XPath expression examples ............................................................................. 21
XPath Expressions for eProducts ............................................................................................ 27
  eProducts Object Tree ....................................................................................................... 27
  eProducts XPath expression testing .................................................................................. 27
  eProducts XPath expression examples .............................................................................. 28
XPath Expressions for Item Templates .................................................................................. 30
XPath Expressions for Price List Quoting .............................................................................. 31
  Price List Quoting Object Tree .......................................................................................... 31
**EFI Pace Object Model Basics**

When you write XPath expressions, you are essentially creating a path through the object model for EFI Pace. XPath expressions tell EFI Pace, in a manner of speaking, ‘in the object model go to this field, from there jump to this field’s container object, on the container object locate a specific field and get it’s value’.

It is important for you to understand the basics of the interrelationships between EFI Pace fields and their container, ancestor and related objects so you can locate the fields you need to write your XPath expressions.

**Fields and Container Objects**

Every field in EFI Pace carries a value and belongs to a specific container object. The container object is the place in EFI Pace within which a field stores its details. Container objects of the same type are linked together in a hierarchical manner like a family tree (with parent, child and grandchild relationships) through a link field. The link field connects one container object to another container object of the same type by calling the desired container object’s primary key.

The primary key is a common field shared by all objects of the same type. An example of a commonly used primary key in EFI Pace is the ID field. Container objects of different types do not share a primary key field, like two friends who each represent their own unique family trees.

Some fields on an object simply hold data and stand alone. They do not link to any other object and they are not a primary or foreign key field.

**Ancestor Objects**

A series of container objects of the same type that share common primary key fields are known as ancestor objects. Like a family tree, the container objects are referred to according to their unique ancestor relationships within their hierarchy. For example, the link field on one container object identifies it as the child of another object. To view higher-level information about the child object, you must link back to its parent object. Likewise, the primary key field on the parent object identifies it as the parent of the child object. Ancestor objects express more information about a field by displaying other fields of pertinent information.

XPath is a means of selecting the value of a field in the database based on your current position in the database. The system utilizes the parent/child/sibling relationships between objects to find related information.
**Example:** You have a company that has customers. Your customers have jobs in the system and each job can have multiple parts. Each part can have multiple materials, shipments, notes, etc. In this example, the company is the parent of customer (customer is the child of company). Job Shipments and materials are children of a JobPart which is in turn a child of job. This could be represented with the following diagram:

```
Company
 |  
Customer
 |       
Job
 |                 
 JOBPart            JobNote
 | 
 | 
JobMaterial     JobShipment     JobPartItems     JobComponents
```

Based on a relationship in the example, knowing which JobMaterial you are considering, you know which JobPart it is in, which job it is in, and which customer ordered it. You do not know, however, anything about a specific JobShipment because there could be more than one JobShipment, just as there could be more than one JobMaterial on a given JobPart. The objects JobMaterial and JobShipment are siblings which are related to the same JobPart, but there is no specific link from a JobMaterial to a JobShipment.

To understand how XPath works, you need to understand the relationships between the objects you are using. You can then move up or down this family tree.

**Example:** You are in a JobPart and you want to know the name of the customer on the job. You know that you need to go from the JobPart to the Job and then to the Customer object because the customer name field is part of that object. The XPath syntax the system uses to go up the family tree looks like this:

```
job/customer/@customerName
```
The job field tells the system to go up one level of the tree to the object job. The customer field tells the system that you want to go up another level to the customer object. The @ symbol designates that the information in the expression is now a field name instead of an object name. In this case, we have requested that the system retrieve the value for the attribute (field) named CustomerName.

**Important:** Expressions are case sensitive. Moving up the family tree requires that the object name start with a lower case letter.

An alternate syntax is also valid, going up the family tree by one level can be designated with two dots:

`../@customerName`

Therefore, the following expressions are the same:

`job/customer/@customerName`

`../@customerName`

**Related Objects**

The same field can appear on several objects of different types. EFI Pace stores the details for that field only once in the field’s container object. When a field makes a ‘guest’ appearance on a related object of a different type, that field is known as a **foreign key**. In this context, the ‘guest’ field is “foreign” to the related object it appears on, and acts as a bridge between two objects that are members of different family trees.

An easy way to identify foreign keys in EFI Pace is to look for the magnifying glass and binoculars icons. When you click on these icons, EFI Pace jumps from the object you are in to the foreign key field’s container object, which enables you to access all the ancestor objects in a new family tree.

**Example:** The InventoryItem object’s parent is Company, but an inventory item can be related to Customer.
Base Objects

Every expression makes use of a base object. The base object is the starting point for the expression. If you are in the Job object, with a Base Object of job, you can use the expression @description to indicate the job's description. If you want the customer name and the base object is job, use the syntax ../@customerName or customer/@customerName to retrieve the name of the customer. This is different than the example in the “Ancestor Objects” topic because of the difference in the base object.

As you proceed through this guide, you will see the base object that is used in each situation. Knowing the base object allows you to know where to begin.

Data Types

It is important to note the data type you should expect to be returned from the XPath expression. Regardless of the complexity of the expression used, the goal will always be to return a single value – either a number or a string (alpha characters and/or numbers). The value that the system returns is used in different ways, depending on the module in which you are working.
Diagram: Generic Object Model Tree

The diagram below represents a generic view of the relationships between types of fields, container objects, ancestor objects and related objects in the EFI Pace object model.
EFI Pace Object Model Browser

Use the EFI Pace object model browser to research the fields and hierarchical relationships of a particular EFI Pace object.

The EFI Pace Object Model Browser is located inside the EFI Pace application. The browser contains a list of the objects in EFI Pace. For the objects listed, the browser provides you with the following:

- The object’s parent object name (with a link).
- A tab with a list of the object’s child objects (with links).
- A tab with a list of the object’s field names.

To access the Object Model Browser, from the EFI Pace Main menu navigate as follows: Click Administration, then move the mouse over System Configuration, then over Object Model, then click Object Model Browser.

Additionally, you can refer to the following Web sites for related information:

- http://www.w3.org/TR/xpath
- http://www.w3schools.com/Xpath/xpath_functions.asp
Steps to Creating an XPath Expression

Use the following procedure to create an XPath expression in EFI Pace. Refer to the related topic following this procedure for detailed instructions.

**Note:** The example data used in this topic and the related subtopics are for Estimating XPath expressions.

1. Determine what the XPath expression needs to do.
   
   Depending on the module you are using, the value returned by the expression will be used in different ways.
   
   **Example:** In Price List Quoting, you can use one expression to select the line of the pricelist, and an additional expression for the quantity to use to calculate the actual price.

2. Determine field names.
   
   • Using the Object Model Browser.
   
   • Using the Debug Mode.

3. Determine the data type of the result field (varies by module).

4. Determine the relationship of the fields to the base object.

5. Write the expression in simple terms.

6. Write the expression in XPath.

7. Test the XPath expression.

8. Enter the XPath expression in EFI Pace.

9. Use the XPath expression.

**Determine what the XPath expression needs to do**

The first step is to determine what the formula needs to accomplish. Determine what fields hold reference values and what conditions need to be met first. Then determine the formula to arrive at the desired results.

**Example:** To preflight a file, allow 15 minutes to review the job jacket, create a job directory on the storage server, transfer the file into the directory, and open the file in preflight software. Then allow 6 seconds per page to check the preflight.

**Formula:** \(.25 + \text{(number of pages} / 3600)\).

The result is in decimal hours.
Determine field names

There are two ways to determine the names of the fields to use.

The first option is to use the Object Model Browser. To access the Object Model Browser, from the EFI Pace Main menu navigate as follows: Click Administration, then move the mouse over System Configuration, then over Object Model, then click Object Model Browser.

On the Object Model Browser page, select the target object, click on the Fields tab and locate the fields you need.

**Note:** This option displays any calculated fields that may not appear on a screen but will make your statement easier.

The second option is to use the Debug Mode. Open an estimate and view the estimate details. Within the estimate, click Administration and then click Toggle Debug Mode. This displays the names of the fields in the Object Model (dataset). On some screens you will need to hover over the field to show the information.

**Example:** Drill into an operation, in this example, a Prepress Operation. In the label area of the field there is a name. It can appear as [quantity] or as EstimatePrepressOp, expr: @addHours.

**Note:** The difference between a field shown as [quantity] and one shown as @addHours is that [quantity] is a calculated field and is forceable, @addHours is not calculated, but is editable.

**Example:** For the preflight example, the fields we need are [quantity], EstimateQuantity/@numPages, EstimatePart/@numSigs. In the Object Model Browser we find that the calculated field totalPages is:

\[
\text{=} \text{ @numPages } * \text{ @numSigs }
\]

This is the expression to use.
Determine the data type of the result field

1. On the EFI Pace Main menu, click **Administration**, move the mouse over **System Configuration**, then over **Object Model**, and then click **Object Model Browser**.

   The Object Model Browser page appears.

2. On the Object Model Browser page, select the target object, then click on the Fields tab and locate the **Type** of the field you need.

   **Example:** EstimatePrepressOp object; quantity field has a **Type** of **Integer**.

---

Determine the relationship of the fields to the base object

All field references in XPath are relative to the object of the result field.

The following chart is an example of EFI Pace object relationships in the Estimating module.

```
Estimating Objects
   Company
   | Estimate
   | EstimatePart
   | EstimateQuantity
|----------------|------------------|------------------|------------------|------------------|------------------|
```

We refer to these relationships as:

- estimatePart is a **child** of estimate
- estimatePart is the **parent** of estimateQuantity
- estimatePress is a **sibling** of estimatePaper

The notations for these are as follows:

- move up a generation: ../
- move to a sibling: /<object name>
- move down a generation: <field name>[condition]/<object>
Note: The [condition] is a way to select a field of a certain value and find data for that record.
Example: ../EstimatePress[@pressIndicator=0]/@runSizeWidth refers to the sibling object EstimatePress with a specific record where pressIndicator = 0 (primary press) and gets the runSizeWidth.

Write the expression in simple terms

The following is an example of an expression in simple terms.

\[(\frac{1}{4} \text{ hour} + (\text{sum(numPages for all parts)}/3600)) \times 100\]

Write the expression in XPath

The following is an example of the simple expression above in XPath expression.

Note: To sum a value, you must go to the parent and then sum the children.

Tip: You can use the Expression Builder feature in EFI Pace to create and validate the expression directly in the EFI Pace system. Refer to the “Enter the XPath expression in EFI Pace” topic in this Reference Guide for more information.

Tip: While it is not required formatting for all XPath expressions, it is recommended that you enclose expressions in parentheses to reduce chances for unexpected results. Example: (@quantityOverride)

In this example, we go up to the Estimate and then refer to the EstimateParts.

\[(.25 + (\text{sum(../..../EstimatePart/@totalPages})/3600)) \times 100\]

Test the XPath expression

Tip: You can use the Expression Builder feature in EFI Pace to create and validate the expression directly in the EFI Pace system. Refer to the “Enter the XPath expression in EFI Pace” topic in this Reference Guide for more information.

1. On the EFI Pace main menu, click Administration, then move the mouse over System Tools, and then click XPath Evaluator.
   The XPath Evaluator page appears.
2. On the XPath Evaluator page, do all of the following:
• In the **Object Type** field, enter the base object for the expression.

• In the **Primary Key** field, enter the primary key/ID for the record of the object that you want to test. This value is usually the number at the end of a URL (web address) when you are viewing a detail record in EFI Pace.

**Example:**

http://server/estimating/object/FinishingOperation/detail/5001 contains a primary key of 5001.

• In the **Data Type** field, select one of the following types of data that you expect to be returned for the expression.

  - **Integer** - you expect the value to be a whole number (no decimals).
  - **Double** - you expect the value to be a decimal number.
  - **String** – you expect the result to be in numbers and letters.

  **Important:** Selecting **String** enables you to see numeric values returned. If the data type is numeric, and a string is returned, a ‘null’ error is the result.

• In the **Expression** field, enter the XPath expression to test.

3. Click **[Evaluate]** and the Result should produce the value you expect.

### Enter the XPath expression in EFI Pace

1. On the EFI Pace main menu, click **Administration**, then move the mouse over **System Setup**, then over **[Module Name]**, then over **Misc Setup**, and then click **[Module] Expression Setup**.

   The [Module] Expression List page appears.

2. Do one of the following on the [Module] Expression List page:

   • To add a new XPath expression on a separate page, click **[Add New Record]**.

      The [Module] Expression Add page appears.

   • To add a new XPath expression directly in the grid, click **[Add In Grid]**.

      A new blank row appears on the [Module] Expression List page.

   • To modify an existing XPath expression, use the Search and Find features to locate the expression, then click the magnifying glass icon next to the desired expression.

      The [Module] Expression Detail page appears.

4. Copy your expression into the appropriate area, or do all of the following to build the expression directly in EFI Pace:

   - Click the Expression Builder icon (Σ) to launch the Expression Builder.
     The Expression Builder pop-up page appears.
   - On the Expression Builder pop-up page, in the Available Objects pane on the left, click the arrow to expand the list of parent and child objects, then select the applicable object you want to use as the base object.
   - On the Expression Builder pop-up page, in the Attributes pane in the middle of the page, click on the object attribute you want to use, and drag it to the Expression group box.
     **Tip:** Drag and drop each desired attribute one at a time to use multiple attributes in the expression.
   - On the Expression Builder pop-up page, in the Functions pane on the right, click on the object attribute you want to use, and drag it to the Expression text box.
     **Tip:** Drag and drop each desired function one at a time to use multiple functions in the expression.

5. If you are on the Expression Builder pop-up page, do either of the following. Otherwise, skip to step 8.

   - To clear the Expression text box click [Reset].
   - To test the expression, click [Validate].

6. On the Expression Builder pop-up page, to evaluate the XPath expression, in the Evaluate group box, do all of the following:

   - In the **[Base Object Name]** field, enter the primary key for the base object you selected.
   - In the **Type** field, select the data type you want to use to evaluate this XPath expression.
   - Click [Evaluate].
     The result should produce the value you expect.

7. On the Expression Builder pop-up page, do one of the following:

   - To save the expression and close the pop-up page, click [OK].
   - To close the pop-up page without saving the expression, click [Close].
The Expression Builder pop-up page closes.

8. Do one of the following:
   • If you are on the [Module] Expression Add page, to save the new expression, click [Add].
   • If you are on the [Module] Expression List page, or on the [Module] Expression Detail page, to save your changes, click [Update].
   • If you are on the [Module] Expression Detail page, to delete this expression, click [Delete], then click [OK] to confirm the deletion.
   • If you are on the [Module] Expression List page, to delete an expression, check the box in the Delete field next to the desired expression, then click [Update].

**Use an XPath expression**

Use of XPath expressions vary by module, refer to the related module topics in this Reference Guide.
XPath expression advanced formatting

To create advanced XPath expressions, you can refer to the following additional XPath expression formatting tips.

**Tip:** Appendix A: Syntax in this Reference Guide, contains a detailed listing of syntax statements and operators that are currently available.

Refer to the core function library on the W3C® Web site for additional formatting. [http://www.w3.org/TR/xpath#corelib](http://www.w3.org/TR/xpath#corelib)

### Relational Operators

Relational Operators include **or, and, !, <=, <, >=, >**, and **iif(case,then,else)**.

**Important:** The “`iif()`” function is a Pace extension to the XPath language and is NOT part of standard XPath expression language.

**Example:** If sides >= 1 then sides else 1

**Example expression:** `(iif(quoteItem/product/@sides >= 1,quoteItem/product/@sides, 1))`

### Arithmetic Operators

Arithmetic operators include **+, -, *, div, mod(), and sum()**

**Example:** Sum of all product sheets for this quote. The “QuoteProduct[productType/@quickCopyType=2]/@sheets” part of the expression below demonstrates a filter which yields only the QuoteProducts where the product type is a JobPart type.

**Example expression:**

```
(sum(quoteItem/product/quote/QuoteProduct[productType/@quickCopyType=2]/@sheets))
```

### Rounding functions

Rounding functions include the following:

- **round** - returns the closest integer number. If halfway, the system rounds .5 up and .49 down.

  **Example:** If the number is 1.49 or less, ‘round’ takes the number to 1.0. If the number is 1.50 or more, ‘round’ takes the number to 2.0.
**Example expression:** round(quoteItem/product/@sheets)

- **floor** - rounds to negative infinity (rounds down) to next whole number.
  
  **Example:** If the number is 1.25, ‘floor’ rounds to 1.0.
  
  **Example expression:** floor(quoteItem/product/@sheets)

- **ceiling** - rounds to positive infinity (rounds up) to the next whole number.
  
  **Example:** If the number is 1.25, ‘ceiling’ rounds to 2.0.
  
  **Example expression:** ceiling(quoteItem/product/@sheets)
XPath Expressions for Estimating

The following topics contain information related to Estimating expressions.

**Estimating Object Tree**

```
Company
  Estimate
    EstimatePart
    EstimateQuantity
```

- EstimatePrepressOp
- EstimatePress
- EstimateFinishingOp
- EstimateOutsidePurch
- EstimatePaper
- EstimateInk
- EstimateActivity

**Prepress Operations**

**Base Object:** EstimatePrepressOp

The system uses this expression to calculate the **Quantity** field on the estimate prepress operation. It can be a number of proofs, a number of flats, number of colors, etc.

**Finishing Operations**

**Base Object:** EstimateFinishingOp

The system uses this expression to calculate the **Quantity** field on the finishing operation of an estimate. The **Quantity** field on the estimate finishing operation uses whole numbers, so the result of this expression should be rounded up or down (as appropriate) to a whole number.

**Price List Items**

**Base Object:** EstimateItem

The system uses two different expressions based on the estimate item, when determining the pricing on the pricelist that is related to the item. One is for the Product Lookup Quantity and the other is for the Product Pricing Quantity.

Refer to the “XPath Expressions for Price List Quoting” topic in this Reference Guide for more information.
Estimating XPath expression examples

The following are examples of XPath expressions for Estimating.

- For a simple price list: 1-10 @ 1.75, 11-25 @ 1.50, 26 and over @ 1.25.

  **Example expression:**

  \[
  ((iif (../@quantityOrdered <= 10, (../@quantityOrdered * 1.75, iif (../@quantityOrdered >= 11 and ../@quantityOrdered <= 25, (../@quantityOrdered * 1.50, (../@quantityOrdered * 1.25)))))
  \]

- Final Size in 8.5 x 11 equivalents.

  **Example expression:**

  \[
  \frac{(..//../@finalSizeHeight * ..//../@finalSizeWidth \div (../../../../@sizeDenominator)\times (../../../../@sizeDenominator)\times 93.5)}{}
  \]

- Banner Hemming - result is final size perimeter feet.

  **Example expression:**

  \[
  (((..//../@finalSizeHeight \div (../../../../@sizeDenominator \times 12)) + (..//../@finalSizeWidth \div (../../../../@sizeDenominator \times 12))) \times 2 \times ../@quantityOrdered
  \]

- Grommets - for each part, put a grommet along each side. Max span = 4 feet, minimum 4 per piece (each corner).

  **Example expression:**

  \[
  ../@quantityOrdered \times (ceiling((..//../@finalSizeHeight \div (../../../../@sizeDenominator \times 12)) \div 4) \times 2 + ceiling((..//../@finalSizeWidth \div (../../../../@sizeDenominator \times 12)) \div 4) \times 2)
  \]
• Laminating - 28" wide - run inches - add 20% if under 7 point or over 16 point or quantity is under 1000. If the sheet width is over 28" it has to run short edge into the laminator, if under 28" then long edge.

**Example expression:**

\[
\text{iif}(.//\text{EstimatePaper}[1]/\text{paperWeight}/@caliper < .007, 1.2, \\
\text{iif}(.//\text{EstimatePaper}[1]/\text{paperWeight}/@caliper > .016, 1.2, \\
\text{iif}(./@\text{sheetsOffPress} < 1000, 1.2, 1))) * \\
(\text{iif}(.//\text{EstimatePress}[@pressIndicator=0]/@runSizeWidth \text{div} \\
(././././@\text{sizeDenominator})>28, \\
((.///\text{EstimatePress}[@pressIndicator=0]/@runSizeWidth \text{div} \\
(././././@\text{sizeDenominator})), \\
((.///\text{EstimatePress}[@pressIndicator=0]/@runSizeHeight \text{div} \\
(././././@\text{sizeDenominator})))) \\
)) * ./@\text{sheetsOffPress}
\]

• Laminate material IN msi.

**Example expression:**

\[
(.///\text{EstimatePress}[@pressIndicator=0]/@runSizeWidth \text{div} \\
././././@\text{sizeDenominator}) * \\
(.///\text{EstimatePress}[@pressIndicator=0]/@runSizeHeight \text{div} \\
././././@\text{sizeDenominator}) \\
* ./@\text{sheetsOffPress} \text{div} 1000
\]
• Die rule length - run number up from die cutting operation (5005) - fold patterns are for presentation folders and folding cartons. If not one of the defined fold patterns then 1000.

**Example expression:**

```xml
iif(../../@foldPatternKey = "8:41",
   (../EstimateFinishingOp[@finishingOperation=5005]/@numUp) *
   iif(../@spineSize = 0,
      (../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 3 +
      (../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 6 +
      (../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 8 +
      (../@oddPanelWidthSize div../@oddPanelSpineSize * 4),
   iif(../@foldPatternKey = "8:42",
      (../EstimateFinishingOp[@finishingOperation=5005]/@numUp) *
      ((../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 6 +
      (../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 8 +
      (../@oddPanelWidthSize div../@oddPanelSpineSize * 4),
   iif(../@foldPatternKey = "24:41",
      (../EstimateFinishingOp[@finishingOperation=5005]/@numUp) *
      ((../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 6 +
      (../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 8 +
      (../@oddPanelSpineSize div../@oddPanelSpineSize * 20),
   iif(../@foldPatternKey = "24:42",
      (../EstimateFinishingOp[@finishingOperation=5005]/@numUp) *
      ((../@finalSizeHeight div ../@finalSizeWidth div ../@tabFace div ../@tabFoot div../@spineSize * 6 +
```
Color Proofs - Press Forms – Use Press Size.

**Example expression:**

\[
\text{iif(} (.//@colorsSide1 \geq 4 \land ./ EstimatePress/@pressIndicator = 0 \land (.// EstimatePress/@runMethod = 3 \lor .// EstimatePress/@runMethod = 7), (.// EstimatePress/@numPressForms \times 2), .// EstimatePress/@numPressForms) + \\
iif( (./@colorsSide1 \geq 4 \land .// EstimatePress/@pressIndicator = 1, (iif (.// EstimatePress/@runMethod = 3 \lor .// EstimatePress/@runMethod = 7, 2, 1), 0) + \\
iif(.//@colorsSide1 \geq 4 \land .// EstimatePress/@pressIndicator = 2, (.// EstimatePress/@runMethod = 7, 2, 1), 0) + \\
\]

To apply a calculation to the first part of an estimate.

**Example expression:**

\[
iif(.//@id = \text{min}(.// EstimatePart/@id), <\text{value or expression}>, 0)
\]

Proof pricing per square inch for standard sizes.

**Note:** Once an if statement is found true, the system returns the result and ignores the remaining true statements. Example: Even though 4x5 is smaller than all other statements, the first result, 54, is returned.

**Example:**

If the flat size is: \( \leq 6x9 \), use the 6x9

If \( > 6x9 \) and \( \leq 11x14 \), use 11x14
If >11x14 and <= 16x20 use 16x20
If > 16x20 and <=20x24 use 20x24
else use 28x40.

Example expression:

```
    iif(../@flatSizeHeight >= ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 9) and
        (../@flatSizeWidth div ../@sizeDenominator <= 6),54,
    iif(../@flatSizeHeight >= ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 14) and
        (../@flatSizeWidth div ../@sizeDenominator <= 11),154,
    iif(../@flatSizeHeight >= ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 20) and
        (../@flatSizeWidth div ../@sizeDenominator <= 16),320,
    iif(../@flatSizeHeight >= ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 24) and
        (../@flatSizeWidth div ../@sizeDenominator <= 20),480,
    iif(../@flatSizeHeight < ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 9) and
        (../@flatSizeHeight div ../@sizeDenominator <= 6),54,
    iif(../@flatSizeHeight < ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 14) and
        (../@flatSizeHeight div ../@sizeDenominator <= 11),154,
    iif(../@flatSizeHeight < ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 20) and
        (../@flatSizeHeight div ../@sizeDenominator <= 16),320,
    iif(../@flatSizeHeight < ../@flatSizeWidth and
        ../@flatSizeHeight div ../@sizeDenominator <= 24) and
        (../@flatSizeHeight div ../@sizeDenominator <= 20),480,960)
```
• To get the total material cost for an Estimate Part, from a prepressOp or finishingOp.

This shows two conditions, EstimateActivity.estimateQuantity = (../@id) AND EstimateActivity.hours = 0 (null = 0 in an integer field). Return 12% of the total material cost on the part.

**Example expression:**

0.12 * sum(../EstimateActivity[@estimateQuantity=(../@id) and @hours =0]/@cost)
XPath Expressions for eProducts

eProducts uses XPath expressions to change variable data fields. In most cases, the expression type is Caption Resolution.

Additionally, use an XPath expression to specify which eProduct Template to use. In this case, use the Expression Type of Template Resolution.

The XPath expression only changes a field on the ProductOrder or uses a different template when a field length is too long and requires a smaller font size. There are no expressions that affect the ProductQuantity.

Base objects are not required in eProducts, but the field# must be correct.

eProducts Object Tree

```
  Company
   |
   Product
    |
   ProductOrder  ProductQuantity  ProductTemplate
```

eProducts XPath expression testing

Use the “Test the XPath expression” topic in this Reference Guide to test eProducts XPath expressions. Enter the following information in the related fields of the XPath Evaluator tool.

**Important:** Ensure you upload the eProducts template to EFI Pace, have the primary key for eService order, and have previewed the template at least once before you test the eProducts XPath expression.

Do all of the following:

- In the **Object Type** field, enter **ProductOrder**.
- In the **Primary Key** field, enter the eService order primary key.
- In the **Data Type** field, select the data type **String**.
- In the **Expression** field, enter the XPath expression you are testing.
eProducts XPath expression examples

The following are examples of XPath expressions for eProducts.

- If a field is blank, move a field up.
  
  **Example:** To move fields, use the if then else to say: If field 4 is blank, then make this field xyz, or else keep it as it currently exists.
  
  **Example expression:** `iif( null-to-zero(@field4) = 0 , @field6, @field5 )`

- If a prefix needs to be added to a field.
  
  **Example:** Telephone/fax number xxx-xxx-xxxx. To place a letter in front of the phone number, use concat function. The example below looks to see if the field is populated, if it is not, it uses field5. In this example, it looks for the fax number and adds the F in front. If populated, it adds the T in front of field 3 for phone.
  
  **Example expression:** `iif(null-to-zero(@field3) = 0, concat("F ",@field5), concat("T ", @field3)`

- To bring fields together in the same line.
  
  **Example:** Bring 3 fields together as city, state, zip.
  
  **Example expression:** `iif(null-to-zero(@field3) = 0 or null-to-zero(@field4)=0 or null-to-zero(@field5)=0, @field8, concat(@field4, " ", @field5, " ", @field6))`

- To capitalize the entire field.
  
  **Example expression:** `translate(@field1, "abcdefghijklmnopqrstuvwxyz","ABCDEFGHIJKLMNOPQRSTUVWXYZ")`
• You can use XPath to check the length of a specific field, which affects which template the EFI Pace system should use. The example below looks at the string length of two fields together to determine which template to use.

```xml
iif(string-length(@field1) + string-length(@field2) > 44,
   product/ProductTemplate[1]/@productTemplate,
   iif(string-length(@field1) + string-length(@field2) > 35,
       product/ProductTemplate[2]/@productTemplate,
       iif(string-length(@field1) + string-length(@field2) > 28,
           product/ProductTemplate[3]/@productTemplate,
           iif(string-length(@field1) + string-length(@field2) > 24,
               product/ProductTemplate[4]/@productTemplate,
               iif(string-length(@field1) + string-length(@field2) > 21,
                   product/ProductTemplate[5]/@productTemplate,
                   product/ProductTemplate[0]/@productTemplate) ) ) )
```
XPath Expressions for Item Templates

Item templates can be used to set values on the following objects when the item templates are used to create JobParts (as with a PaceConnect). Almost any attribute on each of these objects can have a value defined on the item template. The value can be a static value (same on every job), or one that changes for each (such as a value that changes with the quantity ordered).

The base objects are listed below.

**Base Objects**
- JobComponent
- JobMaterial
- JobNote
- JobPart
- JobShipment
XPath Expressions for Price List Quoting

The following topics contain information related to Price List Quoting expressions.

Price List Quoting Object Tree

```
Company
  |
  Quote
  |
  ------------------
  QuotePrice    QuoteQuantity    QuoteProduct    QuoteShipment
  |
  ----------------------
  QuoteItem    ProductPrice
  |
  QuoteItemPrice
```

Common quote-related EFI Pace objects

**Tip:** To use ePaceStation to view all the fields on an object and to view objects that are related to the QuoteItemPrice object, navigate as follows from the EFI Pace Main menu: click **Administration**, then click **ePaceStation**, at the bottom of the page click **Create Inquiry**, select **QuoteItemPrice** then click **[Submit]**, click **Add Fields**, maximize pop-up page and view object fields OR view the path to a related object from the drop-down list.

From the QuoteItemPrice object you can use any fields in the object. **Examples:** (@quantityOverride) or (@quantityValue)

**Important:** When EFI Pace examines an XPath expression to get a calculated quantity, it views the QuoteItemPrice object as the default system object from which to navigate (it assumes the field name in your expression is on the QuoteItemPrice object). Due to this system logic, you may provide just the QuoteItemPrice object field name in your most basic XPath expression – no “path” to this system object is required.

For EFI Pace to accept an XPath expression for any field on the QuoteItemPrice object, you must format the expression as follows:

```
(@fieldname)
```
From the QuoteItemPrice object, you can navigate to other system objects and fields.

**Examples:** Here are a few of the objects and fields to which you can navigate from the QuoteItemPrice object:

Target Object: QuoteQuantity  
XPath with target field: (quoteQuantity/@quantity)

Target Object: QuoteItem  
XPath with target field: (quoteItem/@quantityMultiplier)

Target Object: QuoteProduct  
XPath with target field: (quoteItem/product/@sheets)

Target Object: Quote  
XPath with target field: (quoteItem/product/quote/@eachOf)

**Other System Objects Formats**

To capture the value for a calculated quantity from any field that does not appear on the QuoteItemPrice object, you must write an XPath expression that tells EFI Pace how to navigate from the QuoteItemPrice object to the field you want on another object (provide the path to the field).

To navigate through the EFI Pace object model hierarchy beyond the QuoteItemPrice object, you may use either of the following formats:

- Identify the field on the current object that links to the field on the next object to which you want EFI Pace to navigate. Enter the current object field name followed by a `/` (forward slash symbol) and then the name of the field on the next object, as follows:  
  (fieldThatLinksToParentObject/@parentField)

  **Example:** (quoteItem/@quantityMultiplier)

- To navigate up an object hierarchy (a family tree), you can use a parent path format. This format makes it explicit in the expression that you are referring to a parent field.

  **Example:** ../@fieldName

**Product Lookup Quantities**

**Base Object:** QuoteItemPrice

The system uses this to select the line of a pricelist to use when calculating the price of the QuoteItem.
Product Pricing Quantities

Base Object: QuoteItemPrice

The system uses this to calculate the extended price for a quote item. This is the quantity that the system multiplies by the unit price before the inventory value and flat price are added to arrive at the final price.

Activity Time Calculations

Base Object: QuoteItemPrice

The system uses this to calculate the number of hours to use for the estimated cost record and the job planning record that will be created when the pricelist quote is converted to a job. This value does not factor into the pricing of the quote, it only comes into play when the quote is converted to a job.

Material Calculations

Base Object: QuoteItemPrice

The system uses this to calculate the quantity of inventory to relate to a QuoteItem. The system uses this quantity in the calculation of the final price for a quote item. The system multiplies this value by a price for the inventory item (determined by the setup of PLQ – Sell Price, Average Cost, Replacement Cost) and then adds it to the flat/unit price calculation. The inventory quantity is also the quantity of material allocated as a JobMaterial when the quote is converted to a job.

Finishing Lookup Quantities

Base Object: QuoteItemPrice

This field is only available in the system if you select the Price List Quoting Use Finishing Part setting. The value of this field is used in the same manner as the Product Look Up Quantity mentioned above with the exception that it is used when the QuoteItem is selected on the Finishing Part of a Price List quote. This expression will typically include some sort of summary function to include all of the parts of the current quote.

Example: A finishing operation of drilling may have its pricing based on the total number of sheets that need to be drilled. Part 1 may have 10 sheets, and part 2 might have 20 sheets, so the desire would be to price the drilling as 30 sheets. The expression would be created accordingly.
**Finishing Pricing Quantities**

*Base Object: QuoteItemPrice*

Similar to the Product Pricing Quantities above with the difference that it is used when the quote item is selected on the Finishing Part of the Quote.

**Estimating Lookup Quantities**

*Base Object: Estimate Item*

Similar to the Lookup Quantities above, the value of this expression is used to select a pricelist line. The base object of this expression is different, however, since it is part of a Cost Based Estimate.

**Estimating Pricing Quantities**

*Base Object: EstimateItem*

Similar to the Product Pricing Quantities above, this value is used to calculate the extended price for the item.
XPath Expressions for System Settings

The following topics contain information related to System Settings expressions.

Job Cost Settings - Default Dept for WIP/COGS integration

**Base Object:** JobCost

If the customer is using GL Departments in their accounting, and they want to have a GL Department designated on the journal entries that are created when JobCosts are posted, the department used will be the one designated by this XPath expression. Locate the setup for this field on the Job Cost Setup screen.

The default for this expression is the GL department of the activity code on the costing record:

```
activityCode/@glDepartment
```

User Defined Fields - Calculated Fields

**Base Object:** The object to which the user-defined field has been added.

User Defined Calculated fields use an XPath expression to calculate a value based on the values of other fields at the time the expression is calculated.

Report Parameters

**Base Object:** Designated on the report definition page.

You can define parameters to pass to a report when it is launched. This allows the report to run without requiring you to enter all of the parameters manually. For example, when a Job Jacket report is run, the system does not prompt you for the job number and/or part number that should be used for the report. The values the system passes to the report are based on the results of the XPath expressions on the report front end. The base object for the XPath expressions used for a given report’s parameters is designated on the report definition page. This is the same base object used to determine the context in which the report will be displayed when the user accesses a report by clicking a report context button in the application.
XPath Expressions for User-defined Defaults

The following topics contain information related to User-defined Defaults expressions.

**On Create expressions (using parent object)**

**Base Object:** The parent of the object being added/created.

The On Create user-defined defaults are assigned when the Add form is called to add a new instance of an object.

**On Persist expressions (using current object)**

**Base Object:** The object that is being created.

The On Persist user-defined defaults are assigned when the object is written to the database.
Appendix A: Syntax

The following syntax statements and operators are currently available.

**Important:** EFI Pace uses XPath 1.1 with custom extensions. If you have an XPath reference guide, it may contain version 2.0 expressions which will not be understood by EFI Pace.

**Note:** Use parentheses to order calculations.

**Tip:** Refer to the core function library on the W3C® Web site for additional formatting. [http://www.w3.org/TR/xpath#corelib](http://www.w3.org/TR/xpath#corelib)

### Arithmetic Operators

The following table contains a list of arithmetic syntax.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Plus</td>
</tr>
<tr>
<td>-</td>
<td>Minus</td>
</tr>
<tr>
<td>*</td>
<td>Multiply by</td>
</tr>
<tr>
<td>div</td>
<td>Divide by</td>
</tr>
<tr>
<td>mod</td>
<td>Modulus</td>
</tr>
<tr>
<td>Null-to-zero</td>
<td>null-to-zero if @addHours returns [null], null-to-zero(@addHours) returns 0</td>
</tr>
</tbody>
</table>

### Comparison operators

The following table contains a list of comparison operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
Comparison operators (continued)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>Contains</td>
<td>List of values contains test value</td>
</tr>
<tr>
<td>not</td>
<td>Reverses the true/false value of its argument</td>
</tr>
</tbody>
</table>

Conditional Statement

The following is a conditional statement, which can be nested.

\[
\text{iif}(\text{<value> condition <value>, <result>, <else result>})
\]

String

The following table contains a list of syntax strings.

<table>
<thead>
<tr>
<th>String</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>number(&lt;string value&gt;)</td>
<td>Result is a number. Use to convert string values before math.</td>
</tr>
<tr>
<td>contains(string_1, string_2)</td>
<td>Returns true if string_1 contains string_2</td>
</tr>
<tr>
<td>concat(&lt;string_1&gt;, &lt;string_2&gt;, &lt;string_3&gt;)</td>
<td>Concatenates strings</td>
</tr>
<tr>
<td>starts-with(&lt;string_1&gt;,&lt;string_2&gt;)</td>
<td>Returns true if string_1 starts with string_2, case sensitive</td>
</tr>
<tr>
<td>string(&lt;value&gt;)</td>
<td>Returns value as string, used to return a number as a string.</td>
</tr>
</tbody>
</table>
String (continued)

<table>
<thead>
<tr>
<th>String</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>string-length(&lt;string&gt;)</td>
<td>Returns number of characters in the string.</td>
</tr>
<tr>
<td>substring(&lt;string&gt;,offset,length)</td>
<td>Returns a sub-string of &lt;length&gt; characters starting at &lt;offset&gt;.</td>
</tr>
<tr>
<td>substring-after(&lt;string_1&gt;,&lt;string_2&gt;)</td>
<td>Returns the part of string_1 that follows the 1st occurrence of string_2.</td>
</tr>
<tr>
<td>substring-before(&lt;string_1&gt;,&lt;string_2&gt;)</td>
<td>Similar to substring-after but returns the part before.</td>
</tr>
<tr>
<td>current-date()</td>
<td>Returns the current date and can be used as a comparison in an if statement.</td>
</tr>
<tr>
<td>current-user()</td>
<td>Returns the current user</td>
</tr>
</tbody>
</table>

Summary

The following table contains a list of summary syntax.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum(&lt;value list&gt;)</td>
<td>Result is the total value.</td>
</tr>
<tr>
<td>floor(&lt;value&gt;)</td>
<td>Result is rounded down toward negative infinity (negative numbers increase in absolute value).</td>
</tr>
<tr>
<td>ceiling(&lt;value&gt;)</td>
<td>Result is rounded up toward positive infinity.</td>
</tr>
<tr>
<td>round(&lt;value&gt;)</td>
<td>Result is rounded. &lt; .5 rounds down &gt;= .5 rounds up</td>
</tr>
<tr>
<td>count(&lt;value&gt;)</td>
<td>Result is number of records.</td>
</tr>
</tbody>
</table>
Syntax Notes

- A minimum or maximum function does not exist to compare different fields, instead use the following format to return minimum value of x or y.
  \[ \text{iif} (x < y, x, y) \]
- The min and max functions in XPath only return the minimum or maximum value within a field.
- There is no case statement so use nested \text{iif} statements.
- There is no variable declaration.
- All statements must be included in a single statement. You cannot return multiple values.
- The first expression that returns a true in a nested \text{iif} statement returns the value and ends the \text{iif}.
- It is like writing: if \( x=y \) then \( x \) end if, if \( y=z \) then \( z \) end if.
- You can insert a comment, if desired, at the end of an XPath expression (for a programmer’s note if needed). The comment must be enclosed in braces \{\}.

Example: \( \text{quoteQuantity/@quantity} \ \{ \text{This is the quantity ordered on the quote.} \} \)