

THE YAWL DATA PERSPECTIVE

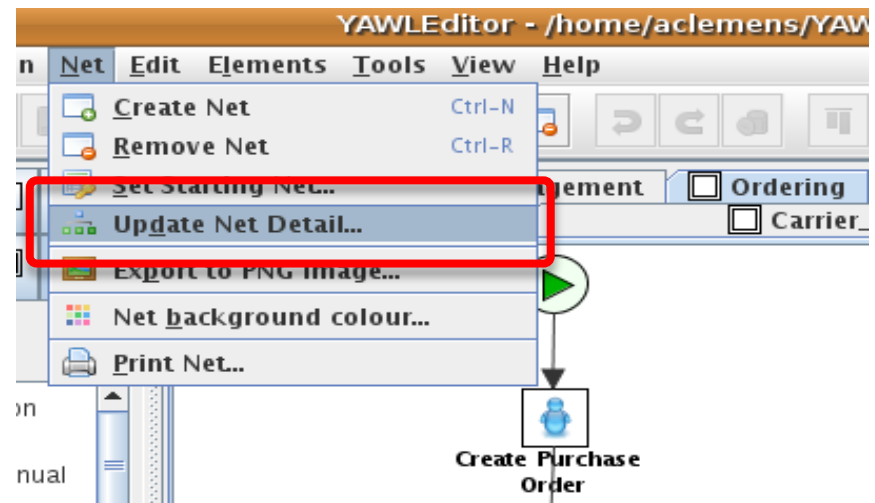
Net and Task Data ("Decompositions")

□ Decompositions Specify

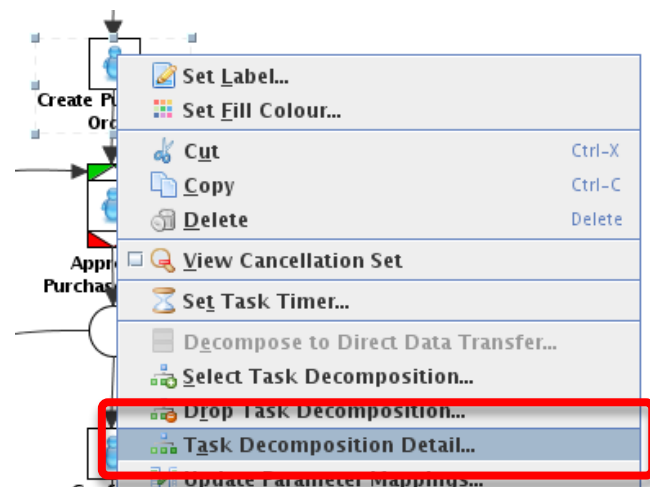
1. Information ("variables") managed at the net- or task-level
 - Name of variable
 - Type of variable
 - Usage of variables (input, output, both)
 - Initial values
2. How information is transferred from net-level to task-level variables at the start of a work item ("input mapping")
3. How information is transferred from task-level to net-level variables at the completion of a work item ("output mappings")

Net and Task Data in the YAWL Editor

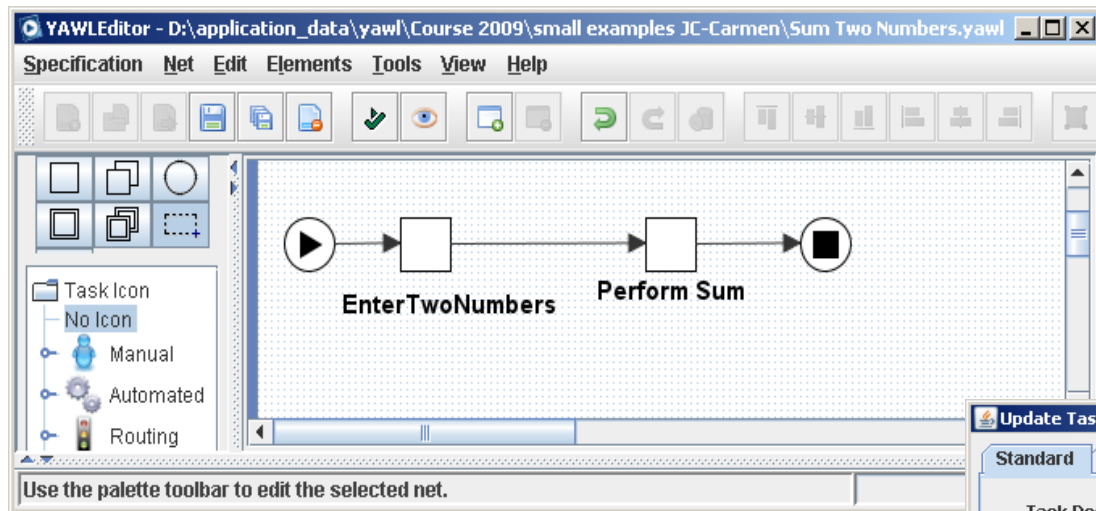
Net
decomposition:
Net → Update
Net Detail...



Task
decomposition:
Context menu of a
task → Task
Decomposition



Net and Task Data in the YAWL Editor



The screenshot shows the "Update Task Decomposition 'Perform_Sum'" dialog box. It has two tabs: "Standard" and "Extended Attributes". The "Standard" tab is active, showing the "Task Decomposition Label" as "Perform_Sum". Below this, the "Task Decomposition Variables" section is highlighted with a red box and labeled "Task Data". It contains a table with the following data:

Name	Type	Usage
s	long	Input & Output

Buttons for "Create...", "Update...", and "Remove..." are located to the right of the table. The "Extended Attributes" tab is also visible, showing "YAWL Registered Service Detail" and "External Interaction" sections.

The screenshot shows the "Update Net Decomposition 'Sum_Two_Numbers'" dialog box. It has a "Net Decomposition Label" field set to "Sum_Two_Numbers". Below this, the "Net Decomposition Variables" section is highlighted with a red box and labeled "Net Data". It contains a table with the following data:

Name	Type	Usage
sum	long	Local
var1	long	Local
var2	long	Local

Buttons for "Create...", "Update...", and "Remove..." are located to the right of the table. At the bottom, there are "Done" and "Cancel" buttons.

XML and YAWL

- When you define a net-level variable (“element”) a net, e.g. variable “sum” for the net called “sum_of_two_numbers”, there is a document in YAWL that looks like this:

```
<sum_of_two_numbers>  
  <sum>13</sum>  
</sum_of_two_numbers>
```

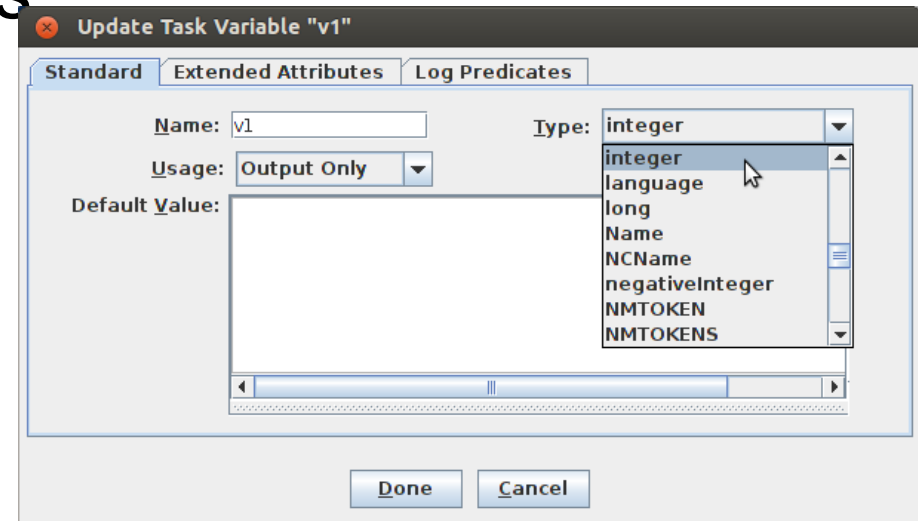
- When you define a task-level variable (“element”) for a task, e.g. variable “S” for a task called “PerformSum”, there is a document in YAWL that looks like this:

```
<PerformSum>  
  <S>9</S>  
</PerformSum>
```

Variable Details

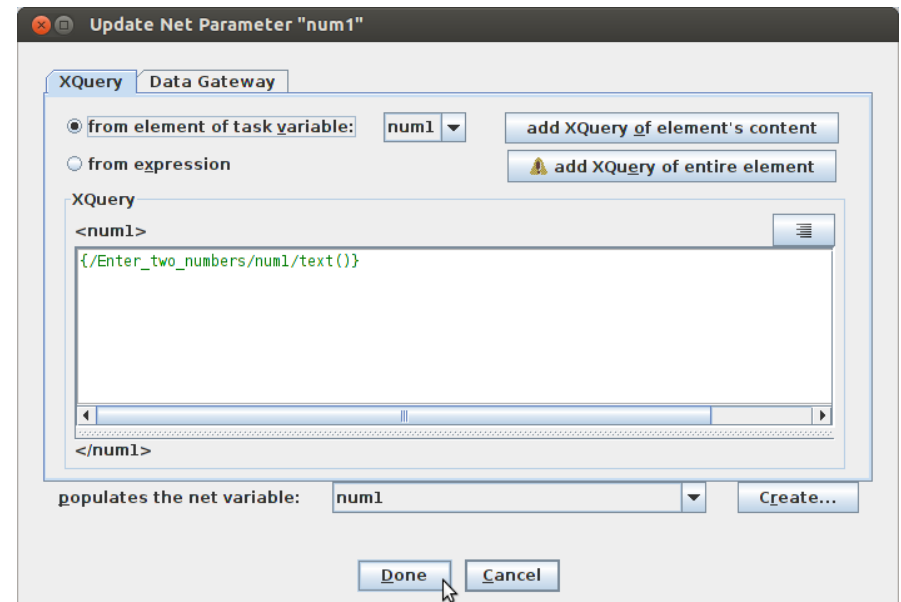
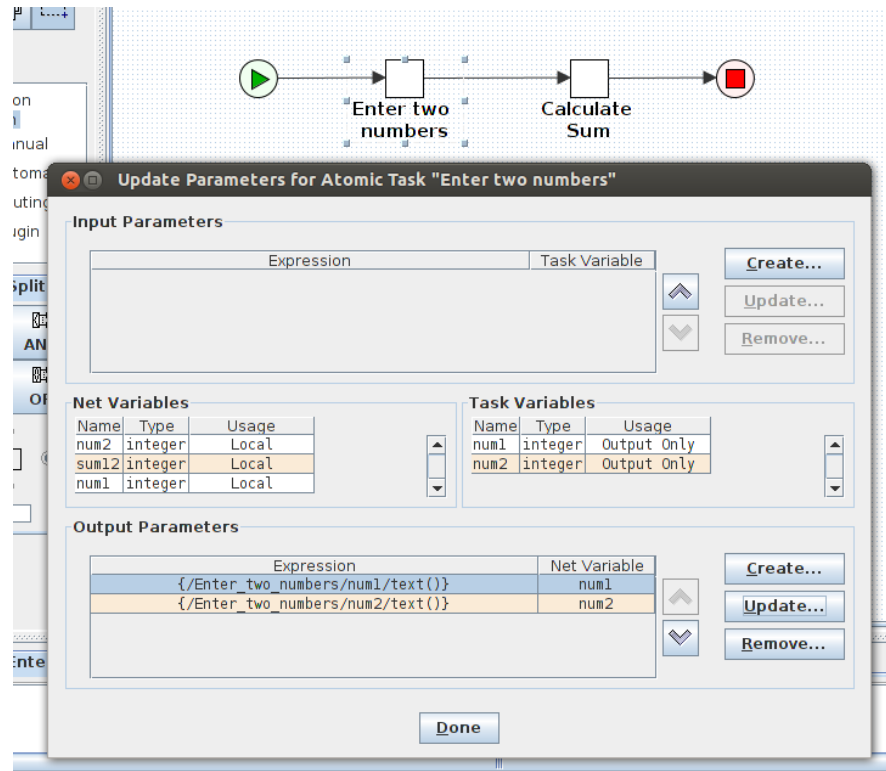
□ Net- and Task-Decomposition dialogs allow to specify

- Decomposition label (unique identifier)
- Variable names
- Data types (simple or complex)
- Type of data usage (input and/or output)
- Default/initial values



Decomposition - Mapping

- In- and output mapping defines how data is transferred between net and task variables
- The mapping is based on XPATH expressions



XML and YAWL

- The *input mapping(s)* for a task specify how the task-level document is created from the net-level document when the task (work item) is *started*. For example, an input mapping like this:

```
<PerformSum>
```

```
<v1>{/sum_of_two_numbers/Variable1/text()}</v1>
```

```
</PerformSum>
```

- selects the content of the net-level element “Variable1” of net “sum_of_two_numbers” into task-level element “v1” of task “PerformSum”.
- Note the XQuery expressions containing simple XPath expressions.

XML and YAWL

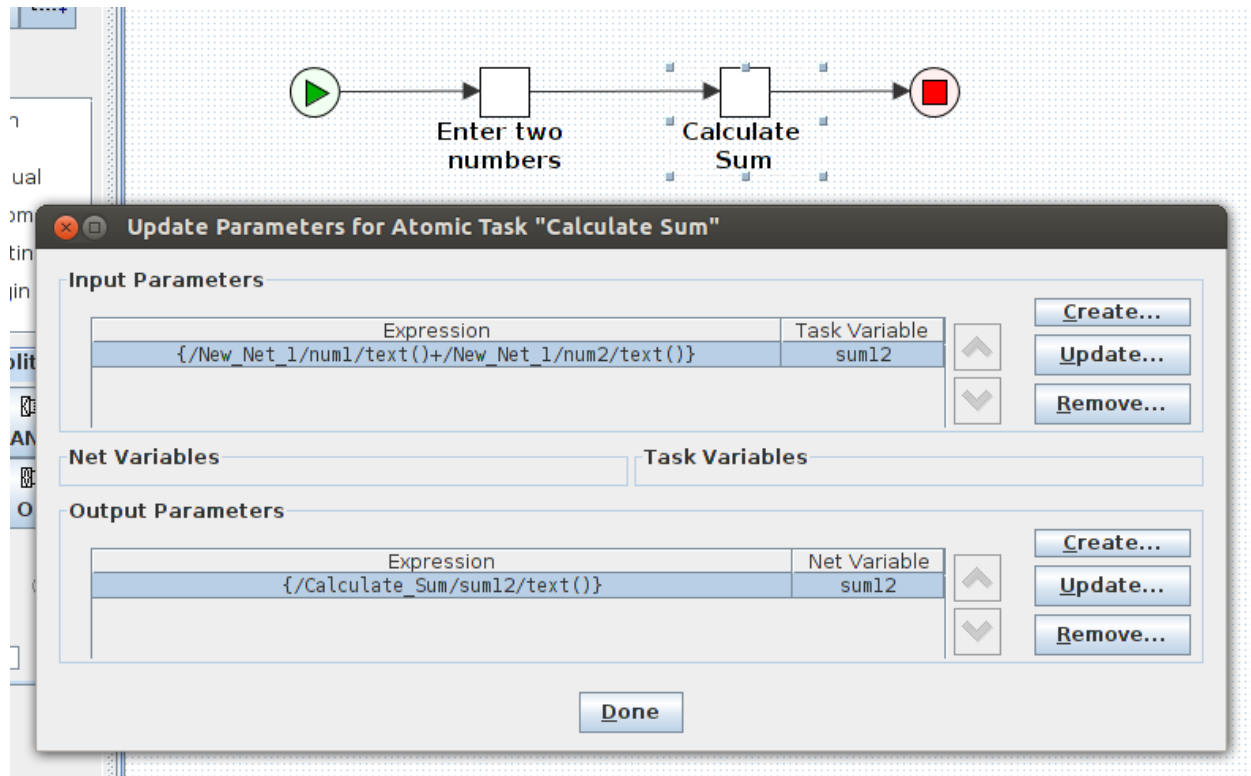
- The output mapping(s) for a task specify how the net-level document is updated from the task-level document when the work item is completed. For example, an output mapping like this:

```
<sum_of_two_numbers>  
  <Variable1>{/EnterTwoNumbers/Var1/text()}</Variable1>  
</sum_of_two_numbers>
```

- selects the content of the task-level element “Var1” of task “EnterTwoNumbers” into net-level element “Variable1” of net “sum_of_two_numbers”.
- Note the XQuery expressions containing simple XPath expressions.

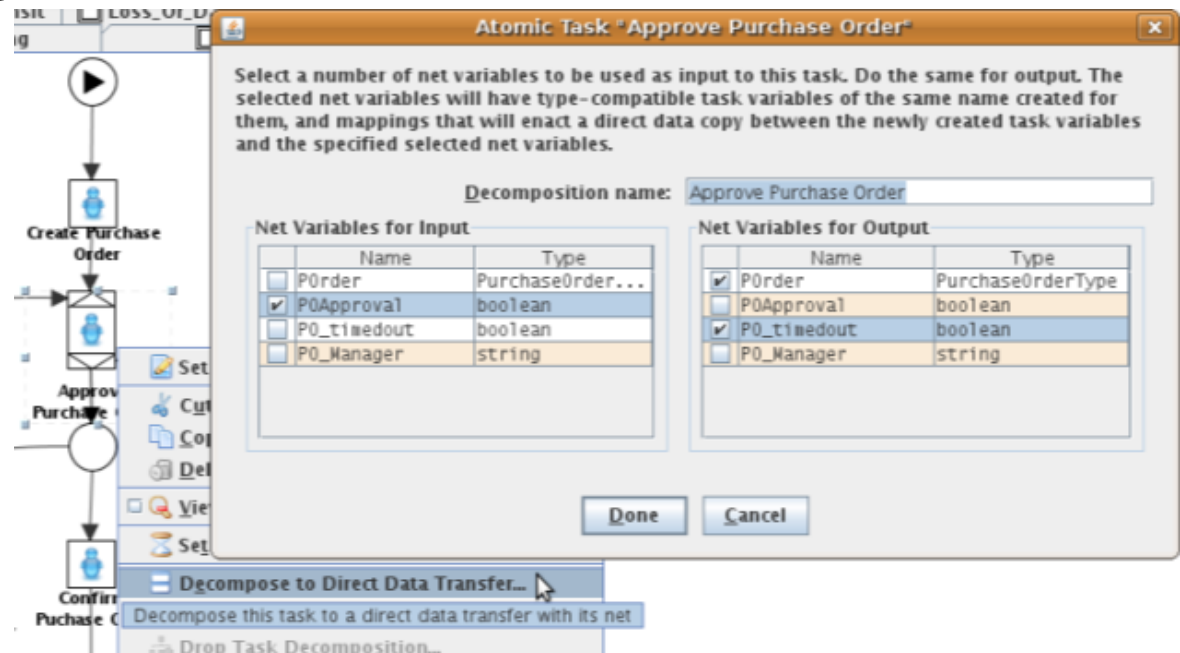
Data Transformation

- As part of the in- and output mapping, data can be transformed



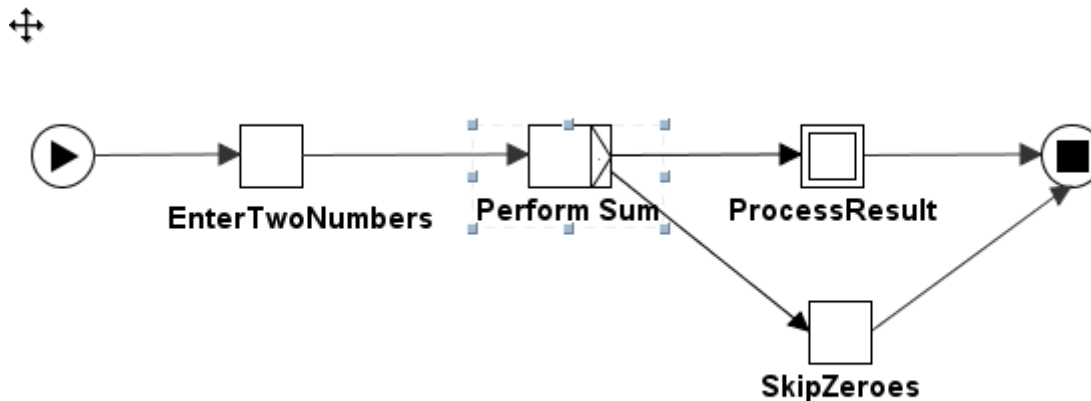
Decomposition – The Lazy Way

- The “Decomposition to direct data transfer” dialogue allows to quickly create a task decomposition
- The respective task should not already have a decomposition
- A set of net variables can be selected and used as template for the task variables
- Variables, data types, in- and output mappings are automatically generated



Data-based Routing in YAWL

For all XOR and OR splits!



Flow detail for Atomic Task "Perform Sum"

Target Task	Predicate
ProcessResult	number(/Sum_Two_Numbers/...
SkipZeroes	number(/Sum_Two_Numbers/...

The bottom-most flow will be used as the default.

Done

Update Flow Predicate

Net variable:

XPath Expression

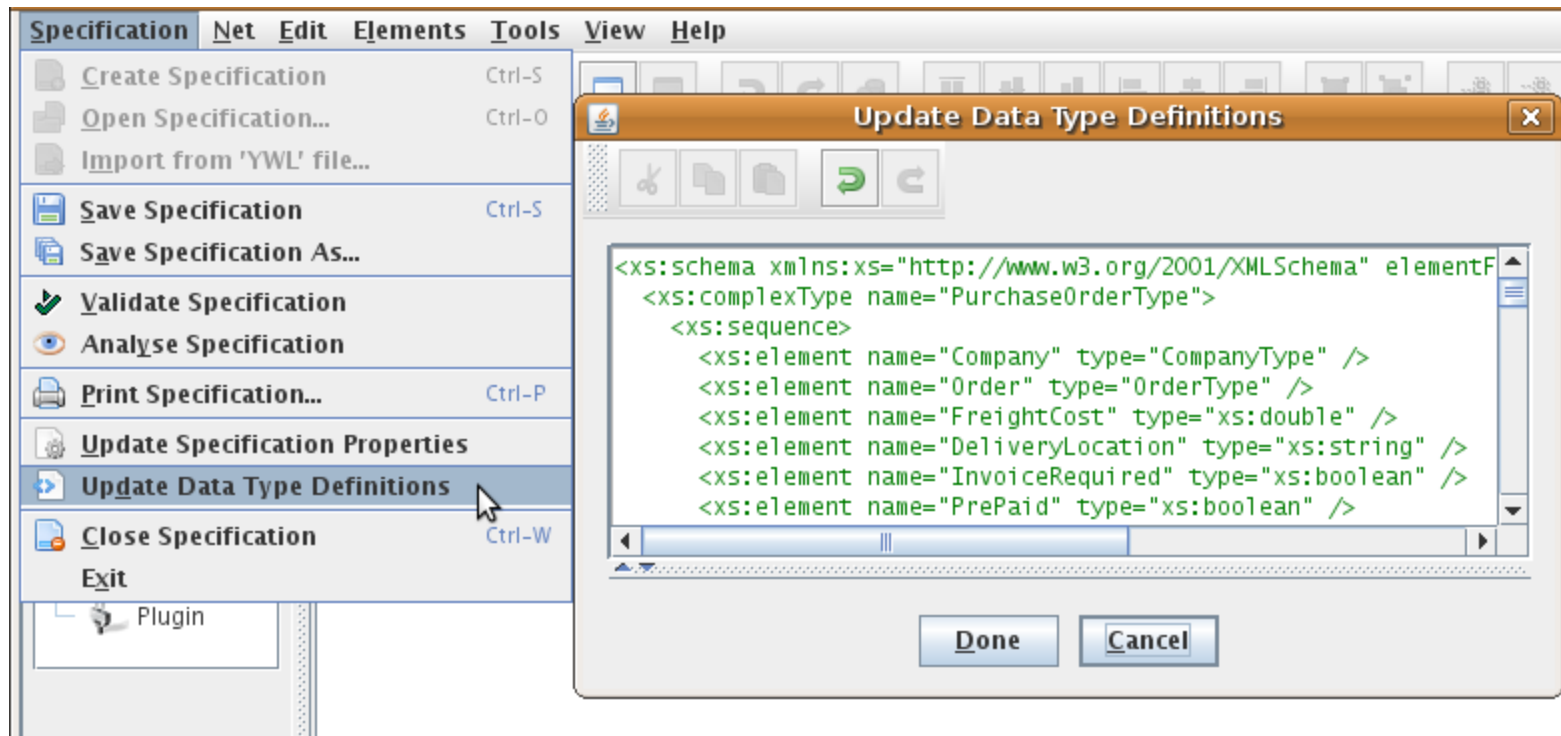
`number(/Sum_Two_Numbers/sum/text()) > 0`

Generates a...

Done Cancel

XML Schema Type Definition

- Editor provides support for
 - Simple XML Schema types
 - Complex XML Schema types



Error Reporting

- ❑ Verification of XQuery and Xpath expressions
- ❑ Suggestion for error resolution

from element of net variable:

XQuery

<POrder>

```
<Compay> <Name/> <Address/> <City/> <State/> <PostCode/> <Phone/> <Fax/> <BusinessNumber/> </Company> <Order> <OrderNumber/> <OrderDate>{fn:current-date()}</OrderDate> <Currency>AUD</Currency> <OrderTerms/> <RevisionNumber>0</RevisionNumber> <Remarks/> <OrderLines> <Line> <LineNumber>1</LineNumber> <UnitCode/> <UnitDescription/> <UnitQuantity>1</UnitQuantity> <Action>Added</Action> </Line> </OrderLines> </Order> <FreightCost>0.00</FreightCost> <DeliveryLocation/> <InvoiceRequired>>false</InvoiceRequired> <PrePaid>>false</PrePaid>
```

End tag </Company> does not match start tag <Compay>

</POrder>

populates the task variable: