Boundless Integrations

The Boundless Integrations XML XPath Functions provide access to additional xpath functions that deal with XML outputs. In addition to being available in assign statements and XSLT's, all Boundless Integrations Custom XPath Functions include public methods for access in a Translation Map.

Prerequisites:

- Update 3rd Party Vendor JAR: xalan.jar. Available free with first Boundless Integrations Custom XPath Function license (extends xslt4j version 2_7_19 to allow 50 custom xpath functions instead of 30)
- Install 3rd Party Vendor JAR: boundless_integrations_core.jar. Available free with first Boundless Integrations license
- BoundlessIntegrations.properties.in: Available with any Boundless Integrations license (reminder: run setupfiles(.sh or .cmd) after placing in the properties directory of each node)

Installation:

- Download boundless_integrations_xpath_xml.jar: After purchasing the XML XPath Functions license, the jar file will be available for download
- Generate, download, and place into the properties directory of each node the BoundlessIntegrations.properties_xmlxpath_ext.in file and the xpathFunctions.properties_boundlessintegrations_xml_ext file
- Stop B2Bi Node(s)
- Place in a location accessible to each node and run bin/install3rdParty(.sh or .cmd)
- Start B2Bi Node(s)

License Update:

- If manually paying for the next year's maintenance, do so.
- After payment has been received (whether automatic or manual), generate, download, and place BoundlessIntegrations.properties_xmlxpath_ext.in file into the properties directory of each node.
- Run bin/setupfile(.sh or .cmd).
- Either bounce each node or execute BNDINT_RefreshCache Business Process.

Get SQL XPath Function:

The Get SQL XPath Function will execute a SELECT statement against any database pool defined in B2Bi. When used in an assign statement or XSLT, it returns a node set with an optionally defined result node and row node. When used in a Translation Map, there are functions to open (get) the pool connection, check the connection status, execute the SQL, get the number of row, getting a specific column from a specific row, and close (free) the pool connection. This allows for additional access to SQL Queries compared with either using a Lightweight JDBC Adapter for business processes and an ODBC Source Translation Map.

Assign/XSLT syntax:

- get-sql(poolName, sqlText [, result [, row]])
  - result defaults to "result"
  - row defaults to "row"

Translation Map methods:

- Object Name: com.boundlessintegrations.xpath.GetSQL
  - No constructor parameters
- openPool(String poolName)
  - Returns a Number value representing an open connection (1 = open)
- isConnected()
  - Returns a Number value representing an open connection (1 = open)
- performSQL(String sqlText)
  - Void function (returns no value)
  - Executes sqlText against poolName and stores result within object
- numRows()
  - Returns a Number value representing the number of rows in result set
- getValue(Number rowNum, String colName)
  - Returns a String value representing the requested row's column
  - Returns a blank value if rowNum is out of bounds
  - Returns a blank value if colName is not a valid column
  - 1-based input (e.g. getValue(1, "FIELD") is the first row
  - colName is always upper case (even for SQL Server)
- closePool()
  - Void function (returns no value)
  - Frees the connection for other uses, but does not clear the result set

Get Document XPath Function:

The Get Document XPath Function will pull a document stored in B2Bi using the DocumentId. When used in an assign statement or XSLT, it returns a node set with the entire document. When used in a Translation Map, there are functions to load the XML document, check for availability of the XML, and to get a String value using an xpath expression. This allows for additional access to XML documents compared with either the DocToDOM function (which is not available within an XSLT) and an XML Source Translation Map.

Assign/XSLT syntax:

- get-doc(DocumentId)
  - DocumentId is the internal Document ID for the document in B2Bi

Translation Map methods:

- Object Name: com.boundlessintegrations.xpath.GetDoc
  - No constructor parameters
- loadXML(String DocumentId)
  - Void function (returns no value)
  - Loads XML from DocumentId and stores result within object
- xmlAvailable()
  - Returns a Number value representing a loaded XML (1 = loaded)
- getString(String xpathExpression)
  - Returns a String value representing the result of the xpathExpression

Get File XPath Function:

The Get File XPath Function will pull a document stored on the file system using the absolute path. When used in an assign statement or XSLT, it returns a node set with the entire document. When used in a Translation Map, there are functions to load the XML document, check for availability of the XML, and to get a String value using an xpath expression. This allows for additional access to XML documents compared with either the DocToDOM function (which is not available within an XSLT and requires the file to be loaded into B2Bi first) and an XML Source Translation Map (which also requires the file to be loaded into B2Bi first).

Assign/XSLT syntax:

- get-file(path)

Translation Map methods:

- Object Name: com.boundlessintegrations.xpath.GetFile
  - No constructor parameters
- loadXML(String path)
  - Void function (returns no value)
  - Loads XML from the path and stores result within object
- xmlAvailable()
  - Returns a Number value representing a loaded XML (1 = loaded)
- getString(String xpathExpression)
  - Returns a String value representing the result of the xpathExpression

Examples:

Get SQL used in an assign to get all pieces with default result and row:

```
<assign to="." from="get-sql('oraclePool', 'select node_name, node_type,
node_status from ops_node_info')" />
```

Process Data:

```
<ProcessData>
  <result>
    <row>
      <NODE_NAME>node1</NODE_NAME>
      <NODE_TYPE>ASI</NODE_TYPE>
      <NODE_STATUS>200</NODE_STATUS>
    </row>
    <row>
      <NODE_NAME>node2</NODE_NAME>
      <NODE_TYPE>ASI</NODE_TYPE>
      <NODE_STATUS>200</NODE_STATUS>
    </row>
  </result>
</ProcessData>
```

Get SQL used in an XSLT loop:

XSLT:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:output method="xml" version="1.0" encoding="UTF-8"
indent="yes"/>
     <xsl:template match="/">
       <Output>
       <xsl:for-each select="get-sql('oraclePool', 'select node_name,
node_type, node_status from ops_node_info')/*">
         <Node>
        <nodeName><xsl:value-of select="NODE_NAME/text()" /></nodeName>
        <nodeType><xsl:value-of select="NODE_TYPE/text()" /></nodeType>
        <nodeStatus><xsl:value-of select="NODE_STATUS/text()"
/></nodeStatus>
      </Node>
       </xsl:for-each>
       </Output>
     </xsl:template>
</xsl:stylesheet>
```

Output Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Output>
  <Node>
    <nodeName>node1</nodeName>
    <nodeType>ASI</nodeType>
    <nodeStatus>200</nodeStatus>
  </Node>
  <Node>
    <nodeName>node2</nodeName>
    <nodeType>ASI</nodeType>
    <nodeStatus>200</nodeStatus>
  </Node>
</Output>
```

NOTE: The asterisk following the get-sql function within the xsl:for-each indicates all child nodes should be pulled. This notation effectively removes the need to specify a result and row when using the get-sql function in an xsl:for-each.

Get SQL used in an XSLT value:

XSLT:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:output method="xml" version="1.0" encoding="UTF-8"
indent="yes"/>
     <xsl:template match="/">
       <Output>
       <xsl:for-each select="Lines/Line">
         <Line>
        <item><xsl:value-of select="item/text()" /></item>
        <qty><xsl:value-of select="qty/text()" /></qty>
        <price><xsl:value-of select="get-sql(&quot;ERP&quot;,
concat(&quot;select price from pricing where item = &apos;&quot;,
item/text(), &quot;&apos; and effdate &lt;= sysdate order by effdate
desc&quot;))/row[1]/PRICE/text()" /></price>
      </Line>
       </xsl:for-each>
       </Output>
     </xsl:template>
</xsl:stylesheet>
```

Output Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Output>
  <Line>
    <item>abc123</item>
    <qty>5</qty>
    <price>14.95</price>
  </Line>
  <Line>
    <item>def456</item>
    <qty>12</qty>
    <price>35.5</price>
  </Line>
  <Line>
    <item>ghi789</item>
    <qty>9</qty>
    <price />
  </Line>
</Output>
```

Get SQL in a Translation Map to get price:

Pre-Session Extended Rule:

```
object getSQL;
integer isOpen;

getSQL = new ("com.boundlessintegrations.xpath.GetSQL");
isOpen = getSQL.openPool("ERP");
```

Extended Rule on #ITEM:

```
string[50] price;

price = "0";

if isOpen = 1 then
begin
  getSQL.performSQL("select price from pricing where item = '" + #ITEM +
"' and effdate <= sysdate order by effdate desc");

  if (getSQL.numRows() >= 1) then price = getSQL.getValue(1, "PRICE");
end

#PRICE = aton(price);
```

Post-Session Extended Rule:

```
getSQL.closePool();
```

Get Document used in an assign to get the full document:

```
<assign to="." from="get-doc('123bdac2453node1')" />
```

Process Data:

```
<ProcessData>
  <Root>
    <order>
      <cust>Adam</cust>
      <date>20140922</date>
      <po numeric="true">123456</po>
      <lines cnt="2">
        <line>
          <item>PBNJ</item>
          <qty>2</qty>
        </line>
        <line>
          <item>PDQ</item>
          <qty>10</qty>
        </line>
      </lines>
    </order>
    <order>
      <cust>Heather</cust>
      <date>20140923</date>
      <po numeric="false">HLC123456</po>
      <lines cnt="0" />
    </order>
    <order>
      <cust>Topher</cust>
      <date>20140924</date>
      <po numeric="true">654321</po>
      <lines cnt="1">
        <line>
          <item>Zombie</item>
          <qty>2000000</qty>
        </line>
      </lines>
    </order>
  </Root>
</ProcessData>
```

Get Document used in an assign to get the second order:

```
<assign to="." from="get-doc('123bdac2453node1')/order[2]/*" />
```

Process Data:

```
<ProcessData>
  <cust>Heather</cust>
  <date>20140923</date>
  <po numeric="false">HLC123456</po>
  <lines cnt="0" />
</ProcessData>
```

Get Document used in an assign to get the order count:

```
<assign to="orderCount" from="count(get-doc('123bdac2453node1')/*)" />
```

Process Data:

```
<ProcessData>
  <orderCount>3</orderCount>
</ProcessData>
```

Get Document used in an XSLT loop:

XSLT:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:output method="xml" version="1.0" encoding="UTF-8"
indent="yes"/>
     <xsl:template match="/">
       <Output>
       <xsl:for-each select="get-doc('123bdac2453node1')/*">
         <Order>
        <po><xsl:value-of select="po/text()" /></po>
        <lines><xsl:value-of select="string(lines/@cnt)" /></lines>
      </Order>
       </xsl:for-each>
       </Output>
     </xsl:template>
</xsl:stylesheet>
```

Output Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Output>
  <Order>
    <po>123456</po>
    <lines>2</lines>
  </Order>
  <Order>
    <po>HLC123456</po>
    <lines>0</lines>
  </Order>
  <Order>
    <po>654321</po>
    <lines>1</lines>
  </Order>
</Output>
```

Get Document in a Translation Map to get line count:

Pre-Session Extended Rule:

```
object getDoc;
string[50] getCount;
integer isAvail, lineCount;

getDoc = new ("com.boundlessintegrations.xpath.GetDoc");
getDoc.loadXML("123bdac2453node1");
isAvail = getDoc.xmlAvailable();

getCount = "0";

if isAvail = 1 then getCount = getDoc.getString("count(//line)");

lineCount = aton(getCount);
```

Get File used in an assign to get the full document:

```
<assign to="." from="get-file('/b2b/share/out/orders.xml')" />
```

Process Data:

```
<ProcessData>
  <Root>
    <order>
      <cust>Adam</cust>
      <date>20140922</date>
      <po numeric="true">123456</po>
      <lines cnt="2">
        <line>
          <item>PBNJ</item>
          <qty>2</qty>
        </line>
        <line>
          <item>PDQ</item>
          <qty>10</qty>
        </line>
      </lines>
    </order>
    <order>
      <cust>Heather</cust>
      <date>20140923</date>
      <po numeric="false">HLC123456</po>
      <lines cnt="0" />
    </order>
    <order>
      <cust>Topher</cust>
      <date>20140924</date>
      <po numeric="true">654321</po>
      <lines cnt="1">
        <line>
          <item>Zombie</item>
          <qty>2000000</qty>
        </line>
      </lines>
    </order>
  </Root>
</ProcessData>
```

Get Document used in an assign to get the second order:

```
<assign to="." from=" get-file('/b2b/share/out/orders.xml')/order[2]/*" />
```

Process Data:

```
<ProcessData>
  <cust>Heather</cust>
  <date>20140923</date>
  <po numeric="false">HLC123456</po>
  <lines cnt="0" />
</ProcessData>
```

Get Document used in an assign to get the order count:

```
<assign to="orderCount"
        from="count(get-file('/b2b/share/out/orders.xml')/*)" />
```

Process Data:

```
<ProcessData>
  <orderCount>3</orderCount>
</ProcessData>
```

Get Document used in an XSLT loop:

XSLT:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:output method="xml" version="1.0" encoding="UTF-8"
indent="yes"/>
     <xsl:template match="/">
       <Output>
       <xsl:for-each select=" get-file('/b2b/share/out/orders.xml')/*">
         <Order>
        <po><xsl:value-of select="po/text()" /></po>
        <lines><xsl:value-of select="string(lines/@cnt)" /></lines>
      </Order>
       </xsl:for-each>
       </Output>
     </xsl:template>
</xsl:stylesheet>
```

Output Document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Output>
  <Order>
    <po>123456</po>
    <lines>2</lines>
  </Order>
  <Order>
    <po>HLC123456</po>
    <lines>0</lines>
  </Order>
  <Order>
    <po>654321</po>
    <lines>1</lines>
  </Order>
</Output>
```

Get Document in a Translation Map to get line count:

Pre-Session Extended Rule:

```
object getDoc;
string[50] getCount;
integer isAvail, lineCount;

getDoc = new ("com.boundlessintegrations.xpath.GetDoc");
getDoc.loadXML("/b2b/share/out/orders.xml");
isAvail = getDoc.xmlAvailable();

getCount = "0";

if isAvail = 1 then getCount = getDoc.getString("count(//line)");

lineCount = aton(getCount);
```

Troubleshooting:

- XML xpath function throws an exception.
  - Check system.log for details. Open an incident with Boundless Integrations if needed.
- All functions return a blank value.
  - Ensure license is up to date.

Support:

Contact Boundless Integrations for any support need. As part of your license, the following support is included at no additional charge:

- Bug fixes
- Installation problems (not including rework of any existing assets)
- Enhancements requests
- General "How Do I" questions

The following services (and any others not explicitly listed above) are available as contract support billed by hour plus any expenses:

- Rework existing business processes/maps/XSLTs
- Custom business processes/maps/XSLTs

Copyright Notice: