Business Objects

Crystal Reports XI

Structuring XML Data for the Crystal Reports XI Native XML Driver

Overview

This document provides suggestions to known issues in the use of the Crystal Reports Native XML driver. It provides sample scripting and illustrates the best methods to structure your XML schemas and XML instances. This document applies to Crystal Reports XI and later.

Contents

Introduction

The Crystal Reports Native XML driver was introduced in Crystal Reports version 10. How your XML schemas and XML instances are structured has a powerful impact on report processing speed when using that driver.

The driver has some limitations and known issues. These issues have been assigned Track ID ADAPT00594769 and ADAPT00586094. Limitations are discussed towards the end of this document.

Important Facts and Suggestions

Installing the CR XML Native Driver

New to Crystal Reports 10 is the Native XML driver. With Crystal Reports XI, the Native XML driver automatically installs with a complete product installation. With Crystal Reports 10, you can download the Native XML driver by clicking **Download Windows JDBC, XML and DB2 Unicode drivers** at

http://www.businessobjects.com/products/downloadcenter/crystalreports.asp

After downloading, consult the included documentation (*install.pdf* and *new_drivers.pdf*) for installation assistance.

Validate XML instance against schemas

The Crystal Reports Native XML driver will not validate XML instances against schemas. Therefore, before you begin, ensure that the XML is valid and without errors. Opening the XML as a local XML file through an Internet Explorer browser 5.5 and higher does not guarantee that the XML has a valid schema or structure. For this reason, it is strongly recommended that you test the XML.

You can download XML Validator, a Microsoft utility, from their support site:

http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/samples/internet/xml/xml_validator/default.asp

The XML Validator will verify that the XML is well formed, valid and error free. Also, this utility indicates elements such as parent/child tags and any attributes associated to elements within the XML and the XML hierarchy structure. You will be able to identify any Table or Row hints for the CR XML driver to use. Table and Row hints are discussed later in this document.

The XML Validator can be used with both XML schemas and Data Type Definitions (DTDs), allows you to verify unsecured URL XML, and to paste any particular XML to test.

Steps to Test the XML

1. Click the link Microsoft's site to the XML Validator provided above.
2. Click the Demo link. The XML Validator appears. (If available, you are able to download the utility as well.)
3. Enter the XML path or URL path. Or, click the **Paste** button to enter the XML.
4. Ensure that the **Validation** check box is selected.
5. Click the **Validate** button.
6. Correct the XML if an error message appears.
7. Repeat the steps if necessary.

Upon completing these steps, you will receive a message stating, "Your XML is well formed and is validated" or an error message related to the XML.

Below the message "Your XML is well formed and is validated", there is parser information for the XML and any parent/child tags or elements sorted by hierarchy. If any attributes are found, this will also be listed according to hierarchy level. This information indicates the XML version and if the XML is Unicode compliant. Based on this information, you can identify any Table or Row hints.

JVM memory issues with large XML files

If your XML file is large or you are experiencing Java Virtual Machine (JVM) memory issues, you can increase the JVM memory size. The Crystal Reports Native XML driver is Java-based and executes in a Java environment. The default JVM memory size is typically 32MB-64MB. You can increase it by changing the Crystal Reports configuration of the JVM in the file, CRConfig.XML.

To increase the memory size to 512MB – 1024MB, modify the following entries in CRConfig.XML:

```
<JVMMaxHeap>1024000000</JVMMaxHeap>
<JVMMinHeap>512000000</JVMMinHeap>
```

In Crystal Reports XI, the default location for CRConfig.XML is:
C:\Program Files\Common\Business Objects\3.0\java

In Crystal Reports XI Release 2, the default location for CRConfig.XML is:

C:\Program Files\Business Objects\Common\3.5\java

Out-of-memory error with a large XML instance

If your XML instance is large, you might encounter an out-of-memory error. To resolve the error message, increase the JVM memory size as mentioned above.

Alternatively, you can organize your XML schema in blocks. This way the Crystal Reports Native XML driver consumes little memory while processing data in streaming mode. In the following example, element **Title** and **Author** are structured in one block, following the recommended structure.

Example 1:

```
<Library>
    <Book>
        <Title>Pride and Prejudice</Title>
        <Author>
            <First Name>
                Jane
            </Last Name>
            </Last Name>
                Austen
            </Last Name>
        </Author>

    </Book>
    ……
    ……
    ……
    <Book>
        <Title>Great Expectations</Title>
        <Author>
            <First Name>
                Charles
            </First Name>
            <Last Name>
                Dickens
            </Last Name>
        </Author>
    </Book>
</Library>
```

In Example 2 a new element, **Library Info**, is added. Since the driver can not know in advance if the Library Info element refers to a Book element, it can not process the file in streaming mode. Depending on the size of the XML data, this may cause performance degradation.

Example 2:

```
<Library>
    <Book>
        <Title>Pride and Prejudice</Title>
        <Author>
            <First Name>
                Charles
            </First Name>
            <Last Name>
                Dickens
            </Last Name>
        </Author>
    </Book>
    ……
    ……
    ……
    <Book>
        <Title>Great Expectations</Title>
        <Author>
            <First Name>
                Charles
            </First Name>
            <Last Name>
                Dickens
            </Last Name>
        </Author>
    </Book>
    <Library Info>
        <Name>
            Shanghai Library
        </Name>
        <Total Circulation>
            130,000,000
        </Total Circulation>
    <Library Info>
</Library>
```

How much memory the Native XML driver will consume is proportional to how much of the XML file the driver has to read in order to create each report record detail.

For example, if a user selects one table in the XML instance, even if the XML instance is very large, Native XML driver will only consume a small amount of memory.

However, if you select several tables which are widely distributed, for example, one at the top of the XML instance, several in the middle, and one at the bottom, the Native XML driver has to read the entire XML instance to generate one record. This can cause the Native XML driver to use more memory resulting in performance degradation.

In the following example, if a user selects one book record in the library, the Native XML driver will consume a small amount of memory, even if the XML instance is quite large. However, if a user selects **Library Name**, several **Book** and **Total Circulation**, the Native XML driver will consume a larger amount of memory relative to the size of the XML instance. You may encounter an out-of-memory error.

Example 3:

```
<Library>
    < Library Name>
        Shanghai Library
    </Library Name>
    <Book>
        <Title>Design UK</Title>
        <Author>
            <First Name>
                Max
            </First Name>
            <Last Name>
                Fraser
            </Last name>
        <Author>
    </Book>
    …
    <Book>
        <Title>The Diary Of A Young Girl</Title>
        <Author>
            <First Name>
                Anne
            </First Name>
            <Last Name>
                Frank
            </Last name>
        <Author>
    </Book>
    …
    <Book>
        <Title>BILLY BUDD SAILOR</Title>
        <Author>
            <First Name>
                Herman
            </First Name>
            <Last Name>
                Melville
            </Last name>
        <Author>
    </Book>
    <Total Circulation>
        130,000,000
    </Total Circulation>
</Library>
```

Avoid many-to-many relationships in your XML schema

For performance considerations, avoid many-to-many relationships in your XML schema.

In a many-to-many relationship, one record in either table can relate to many records in the other table. For example, in a company, the sales agents and products can have a many-to-many relationship, since each agent may sell all products. Conversely, each product can be sold by any agent. Example 4 illustrates this relationship.

Example 4:

```
<Company>
        <Sales Agent>
                <Location>Beijing </Location>
        </Sales Agent>
        <Sales Agent>
                <Location>Shanghai </Location>
        </Sales Agent>
        <Product>
                <Name>Refrigerator </Name>
                <Manufacturer>
                        <Name>
                                Haier
                        </Name>
                        <Contact Person>
                                <Name>
                                        Mark Williams
                                </Name>
                                <Phone Number>
                                        +862151525354
                                </Phone Number>
                        </Contact Person>
                </Manufacturer>
        </Product>
        <Product>
                <Name>TV</Name>
                <Manufacturer>
                        <Name>
                                CHONGHONG
                        </Name>
                        <Contact Person>
                                <Name>
                                        Steve Nash
                                </Name>
                                <Phone Number>
                                        +862188888888
                                </Phone Number>
                        </Contact Person>
                </Manufacturer>
```

```
        </Product>
</Company>
```

Example 4 can be modified to be more efficient. Using Example 5 will reduce the demand on your computer's memory.

Example 5:

```
<Company>
        <Sales Agent>
                <Location>Beijing</Location>
                <Product>
                        <Name> Refrigerator </Name>
                        <Manufacturer>
                                <Name>
                                        Haier
                                </Name>
                                <Contact Person>
                                        <Name>
                                                Mark Williams
                                        </Name>
                                        <Phone Number>
                                                +862151525354
                                        </Phone Number>
                                </Contact Person>
                        </Manufacturer>
                </Product>
                <Product>
                        <Name>TV</Name>
                        <Manufacturer>
                                <Name>
                                        CHONGHONG
                                </Name>
                                <Contact Person>
                                        <Name>
                                                Steve Nash
                                        </Name>
                                        <Phone Number>
                                                +862188888888
                                        </Phone Number>
                                </Contact Person>
                        </Manufacturer>
                </Product>
        </Sales Agent>
        <Sales Agent>
                <Location>Shanghai</Location>
                <Product>
                        <Name> Refrigerator </Name>
                        <Manufacturer>
                                <Name>
                                        Haier
                                </Name>
```

```
                        <Contact Person>
                            <Name>
                                Mark Williams
                            </Name>
                            <Phone Number>
                                +862151525354
                            </Phone Number>
                        </Contact Person>
                    </Manufacturer>
                </Product>
                <Product>
                    <Name>TV</Name>
                    <Manufacturer>
                        <Name>
                            CHONGHONG
                        </Name>
                        <Contact Person>
                            <Name>
                                Steve Nash
                            </Name>
                            <Phone Number>
                                +862188888888
                            </Phone Number>
                        </Contact Person>
                    </Manufacturer>
                </Product>
            </Sales Agent>
</Company>
```

Native XML driver performance issue with a large XML instance

The Native XML driver may encounter a performance issue when the XML instance is large. For example, it might take 10 minutes to extract data from a 20 MB file. It is recommended that you structure your data using multiple XML files, rather than putting everything in a single XML file.

A general guideline is best to put unrelated data from your business view into separate XML files. If the data is related, you can put them in one single XML file and try to structure them in a hierarchy to avoid unrelated join issues (many-to-many relationship) as described above.

Ideally, streaming-style processing is the goal for every XML instance. If various sets of data are included in a single file such that XML driver can not process the file in a single pass, it is recommended that the data be broken out into separate files. If the Native XML driver has to move back and fourth to process data within an XML file, performance will suffer.

Too much data in one XML file risks causing unrelated join issues. The joins within one XML file can also prevent efficient streaming-style processing of the data if not structured well.

Known Issues and Limitations

Avoid creating cyclic references in your schema

There are two cyclic reference scenarios. One occurs when an XML schema element refers to its parent element or grandparent element. This is known as a cyclic reference in the schema. The second scenario occurs when the deepest level of XML schema is more than 100; the Native XML driver will regard it as cyclic reference.

Including or importing a local file

If you want to include or import a local file, begin your URL with "file:///". For example, write your "include" element as follows:

< include schemaLocation="file:///C:/MyDocs/ipo.xsd" >

Special character restrictions

Because they are handled specially by the Native XML driver, do not use the following special characters to define element types and attributes in your XML schema:".", "/", "\", ":" "@".

Inline schema

Inline Schema support is introduced in Crystal Reports XI Release 2. Do not use inline schema in your XML file for versions prior to Crystal Reports XI Release 2.

Inline schemas are XML schema definitions included inside XML instance documents. When used properly, they can validate that the rest of the XML matches the schema constraints in the same way that external schema documents can be used. Likewise, the syntax and semantics of inline schemas are the same as for external schemas.

Inline schemas can be useful in a number of situations, including:

- It is difficult to access external files or URLs for security or platform reasons.
- There is too much diversity in the set of schemas and instances that a system must process, so it is easier to keep the schema as an integral part of the XML document.

The following is an example of using an inline schema:

Example 6:

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns:inl="http://inline">
        <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
        targetNamespace=http://inline xmlns=http://inline elementFormDefault="qualified"
        attributeFormDefault="unqualified">
                <xs:element name="parent">
                        <xs:complexType>
                                <xs:sequence>
                                        <xs:element name="child" type="xs:string"/>
                                </xs:sequence>
                        </xs:complexType>
                </xs:element>
        </xs:schema>
        <inl:parent>
                <inl:child>text</inl:child>
        </inl:parent>
</root>
```

XML from Microsoft ADO is not supported prior to Crystal Reports XI Release 2

In Crystal Reports XI Release 2, users can provide an ADO-format XML file or ADO-format XML stream to the Native XML driver. For XML stream support, users can specify an HTTP URL as a data source in the Native XML driver. From the URL, which links to a servlet, an ASP page, a JSP page or another type of dynamically created Web page, you can provide an XML stream.

ADO XML support is different from XML support in ADO.NET. ADO classic's XML support uses a Microsoft-specific predecessor of XSD (XML Schema Definition) schemas, known as XDR (XML Data Reduced). Unlike ADO classic XML, ADO.NET XML support uses XSD schemas. In the Native XML driver, ADO XML is handled separately, whereas ADO.NET XML is handled the same as other XML which complies with W3C's XML Schema. From more information, refer to Microsoft's online MSDN: http://msdn1.microsoft.com/en-us/default.aspx

Extended type binding not supported by Native XML driver

The Native XML driver does not support extended type binding in XML instances. In Example 7, you define a type **Address** and extend it to **USAddress**. In your XML schema, you define an element **shipTo** of which type is **Address**. But in the XML instance, the Native XML driver does not support binding its type to **USAddress** even if it complies with W3C's schema.

Example 7:

…

```
<!-- Define the base type "Address"-->
<complexType name="Address">
  <sequence>
    <element name="name"   type="string"/>
    <element name="street" type="string"/>
```

```
   <element name="city"   type="string"/>
   </sequence>
</complexType>
…
<!--Define "USAddress" by extending type "Address" -->
<complexType name="USAddress">
 <complexContent>
  <extension base="ipo:Address">
   <sequence>
    <element name="state" type="ipo:USState"/>
    <element name="zip"   type="positiveInteger"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
…
<!— define an element whose type is "Address"-->
<element name="shipTo" type="ipo:Address"/>
…
<!— bind "shipTo" to type "USAddress" -->
        <billTo xsi:type="ipo:US-Address">
                <ipo:name>Robert Smith</ipo:name>
                <ipo:street>8 Oak Avenue</ipo:street>
                <ipo:city>Old Town</ipo:city>
                <ipo:state>AK</ipo:state>
                <ipo:zip>95819</ipo:zip>
        </billTo>
```

Simple elements mapped to a field

If a simple element is mapped to a field by the Native XML driver, and the element occurs continuously in your XML instance, you will get incorrect data if you select that field. The driver will handle the instances as one element. In the following example, the Native XML driver will handle all the **Items** as a single element.

Example 8:

```
<StockItem>
        <Item>A100</Item>
        <Item>A200</Item>
        <Item>A300</Item>
        <Item>A400</Item>
</StockItem>
```

If you select **Items** in your report, you probably want to see the four items formatted the following

Item
A100
A200
A300

A400

However, the result shown in Crystal Reports is:

Item
<Item>A100</Item><Item>A200</Item> <Item>A300</Item><Item>A400</Item>

For more information, search our support site for **ADAPT00594769:** Crystal Reports XI brings in the tag names if the database field has multiple values via XML native connection.

To work around this issue, restructure your XML instance as shown in Example 9.

Example 9:

```
<StockItem>
        <Item>A100</Item>
</StockItem>
<StockItem>
        <Item>A200</Item>
</StockItem>
<StockItem>
        <Item>A300</Item>
</StockItem>
<StockItem>
        <Item>A400</Item>
</StockItem>
```

BusinessObjects Enterprise XI and the Native XML driver

The Native XML driver used in BusinessObjects Enterprise XI does not support different user name and password for XML instance and XML schema documents accessing.

Mixed attributes are not supported in XML schemas

The Native XML driver does not support mixed attributes in XML schemas. For more information, search our support site for **ADAPT00586094**: Error occurs when there is "mixed" attribute in the xml schema.

With **mixed** enabled, the XML Schema provides for the construction of schemas where character data can appear alongside sub elements, and character data is not confined to the deepest sub elements.

The letter to the customer in Example 10 illustrates what a **mixed** attribute means. Characters in orange appear alongside sub elements.

Example 10:

```
<letterBody>
        <salutation>
                Dear Mr.
                <name>Robert Smith </name>.
        </salutation>
        Your order of
        <quantity>1</quantity>
        <productName> Baby Monitor </productName>
        shipped from our warehouse on
        <shipDate> 1999-05-21 </shipDate>. ....
</letterBody>
```

To allow the above example to be valid, you should declared **mixed** as "true" in your XML schema as shown in Example 12.

Example 12:

```
<xsd:element name="letterBody">
 <xsd:complexType mixed="true">
  <xsd:sequence>
   <xsd:element name="salutation">
    <xsd:complexType mixed="true">
     <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="quantity"    type="xsd:positiveInteger"/>
   <xsd:element name="productName" type="xsd:string"/>
   <xsd:element name="shipDate"    type="xsd:date" />
   <! -- Etc. -->
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

The Native XML driver only supports latest XML schema spec

The Native XML driver only supports the latest XML schema specifications. References to older schema specifications should be replaced with the newest version.

For example, if your XML schema refers to http://www.w3.org/2000/10/XMLSchema, change it to http://www.w3.org/2001/XMLSchema.

Defined namespaces in your XML schema

If you have defined namespaces in your XML schema, specify the namespaces in your XML instances. Otherwise, your report can not retrieve data from the XML files that use that schema. Note that no error message appears for this case.

Finding More Information

For more information and resources, refer to the product documentation and visit the support area of the web site at:
http://www.businessobjects.com/

For information about working with ODBC connections to your XML data source, refer to:

http://support.businessobjects.com/communityCS/TechnicalPapers/cr_xml_data_sources.pdf.asp