DB2® Universal Database for OS/390 and z/OS
XML Extender

Release Notes

Version 7

DB2® Universal Database for OS/390 and z/OS
XML Extender

Release Notes

Version 7

Contents

About the Release Notes for DB2® Univeral Database for OS/390 and z/OS, XML Extender Version 7

This document contains information about the DB2 XML Extender 7, supplementing information in *DB2 Universal Database for OS/390 and z/OS® XML Extender Administration and Programming, Version 7*. The information in these Release Notes includes all topics except installation.

Chapter 1. What’s in this document?

This document describes fixes and documentation updates for APAR PQ58249 in PTF UQ65515, that contains updates to XML Extender equivalent to DB2 Universal Database Version 7.2 for Windows and Unix, Fixpak 5.

This document provides updates to information about topics in the *DB2 Universal Database XML Extender Administration and Programming, Version 7*, and the online error messages.

You can find *DB2 Universal Database XML Extender Administration and Programming, Version 7* on the DB2 XML Extender Web site:

`www.ibm.com/software/data/db2/extenders/xmlext/library.html`

Chapter 2. Fixes and updates

The following topics describe fixes provided in this PTF, as well as new and updated information for the *DB2 Universal Database XML Extender Administration and Programming, Version 7* document.

Migrating to the PTF from prior releases

| If you have been using XML Extender Version 7 you must migrate each database
| enabled for XML Extender before using an existing XML-enabled database with
| this PTF. The migration program migrates you to the equivalent of DB2 XML
| Extender Version 7.2 (UNIX/Windows) Fixpak 5 from prior releases.

The following list describes the steps the migration program goes through:

- Creates new stored procedures (dxxGenXMLCLOB and dxxRetrieveXMLCLOB) that return CLOBs
- Alters all of the DB2XML user defined functions (UDFs) to allow you to use the parallel capability for the scalar UDFs
- Creates XMLDBCLOB user-defined types (UDTs) and user-defined functions (UDFs) for use with Unicode and DBCS databases.

To migrate the database: Customize and run the following two jobs in the SDXXJCL data set. Instructions for customizing the jobs are in the JCL prologue for each job.

DXXRBIND
This job rebinds all the DB2 packages and plans.

DXXMIGRA
This job drops, creates, and alters functions and stored procedures.

Failing to do the migration step can cause the **DXXADM disable_server** option and the dxxGenXMLCLOB and dxxRetrieveXMLCLOB stored procedures to fail

Software requirements and set up for the XML Extender administration wizard

Chapter
Administering XML data

Section
Setting up the administration wizard

Update
The following information supplied more detailed information about the correct software requirements and set up for the wizard.

To use the XML Extender administration wizard, you must have your PATH and CLASSPATH environment variables set up properly.

In this section, the following terms are used to indicate product installation directories. These are documentation conventions, not actual variables.

dxx_install

The directory under which the XML Extender has been installed. For example, on Windows operating systems, the default is:

```
c:\dxx
```

db2_install_directory

The directory under which DB2 has been installed. For example, on Windows operating systems, the default is:

```
c:\program files
```

The PATH statement must contain the directories for the Java™ binary files and optionally the XML Extender administration directory. Table 1 describes the requirements for the PATH statement:

Table 1. Path elements

Software	**File Location**	**Path value**
XML Extender administration directory	The dxxadmin subdirectory under *dxx_install*, the directory where you installed XML Extender.	`dxx_install/dxxadmin/` Except on Linux, where the `dxxadmin` directory is called `adm_g`
Java binary files	Depends on the Java package you use, which could be one of the following: • DB2-provided Java Developer's Kit JDK™ or Java Runtime Environment (JRE) • User- installed JDK or JRE And whether you use the **jre** or **java** command.	DB2 JDK: *db2_install_directory* / sqllib / java / jdk / bin
		DB2 JRE: *db2_install_directory* / sqllib / java / jre / bin
		User-installed JDK: *install_directory* / bin
		User-installed JRE: *install_directory* / bin

The CLASSPATH points to the administration wizard jar files, Java class libraries, DB2 JDBC™ driver file, and the Swingall jar file. It is used to invoke the wizard. See the *DB2 Universal Database XML Extender Administration and Programming, Version 7* documentation for more information about starting the wizard.

CLASSPATH syntax:

```
classpath=<dxx_install/dxxadmin/dxxadmin.jar><dxx_install/dxxadmin/<xml4j.jar>;
   <path/java class libraries>;<path/JDBC>;<path/swingall.jar>
```

On Linux, the `dxxadmin` directory is called `adm_g`.

The JDK and JRE are provided with DB2 UDB Version 7, on certain operating systems. When available, the JRE is automatically installed, and the JDK is installed when you choose to install the DB2 UDB Application Development selection. It is optional that you use the JDK or JRE provided with DB2. See DB2 Universal Database: Java Enablement with DB2 Universal Database Version 7.1 Web site for detailed information about Java support in DB2 Universal Database Version 7.1

If you are using an operating system in which Java support is not provided by DB2, you must install the JDK or JRE yourself. Use the link above to the DB2 Java site to find download packages for the various levels of Java that have been tested with DB2. You can also choose to install another version of the JDK or JRE even if you have the DB2–provided versions installed. In this case, you should point to the same JDK, or JRE, installation in both the PATH and in the CLASSPATH environment variables.

Table 2 describes the directories and requirements for the CLASSPATH.

Table 2. CLASSPATH values

Software	**CLASSPATH value**
XML Extender jar files: • dxxadmin.jar • xml4j.jar	• *dxx_install* / dxxadmin / dxxadmin.jar • *dxx_install* / dxxadmin / xml4j.jar On Linux, the `dxxadmin` directory is called `adm_g`.

Java class libraries: determined by whether you are running the **java** or **jre** command.	**java command**
	Point to the `classes.zip` file. For example:
	`db2_install_directory/SQLLIB/java/jdk/lib/classes.zip`
	jre command
	Point to the `rt.jar` file. For example:
	`/db2_install_directory/SQLLIB/java/jre/lib/rt.jar`
	The path will be different if you have installed another JDK or JRE.
	If you more than one version of Java installed on your system, it is important that you point to files in the same version of Java in both the PATH and the CLASSPATH statements.

Swingall.jar file	Available from three sources: • JDK 1.2 or higher, provides Swingall. • DB2 UDB V7 provides Swingall. • You can install Swingall separately, which can be downloaded from the Web. The following example shows the CLASSPATH statement using the DB2–provided Swingall: `db2_install_directory/SQLLIB/java/swingall.jar`
JDBC: `db2java.zip`	Installed with DB2 UDB V7 under: `db2_install_directory/SQLLIB/java/db2java.zip`

XML column

Update user-defined function - redundant line ending characters

A fix is provided for the Update UDF, to prevent an extra line ending character added after the XML declaration and the DOCTYPE declaration.

XML collection

Using DAD files for XML collections

The following section outlines changes that affect how you use DAD files.

SQL composition: using columns with the same name

Selected variables with the same name, even if from diverse tables, must be identified by a unique alias so that every variable in the select clause of the SQL statement is different. The following example shows how you would give unique aliases to columns that have the same names.

```
<SQL_stmt>select o.order_key as oorder_key,key customer_name, customer_email,
p.part_key p.order_key as porder_key, color
qty, price, tax, ship_id, date, mode from order_tab o.
part_tab p ORDER BY order_key, part_key</SQL_stmt>
```

SQL composition: using columns with random values

If a SQL statement in a DAD has a random value, you have to give the random value function an alias in order to use it in the ORDER BY clause. This is because it is not associated with any column in a given table. For example, see the alias for generate_unique at the end of the ORDER BY clause below.

```
<SQL_stmt>select o.order_key, customer_name, customer_email, p.part_key,color,
qty, price, tax, ship_id, date, mode from order_tab o, part_tab p,
table(select substr(char(timestamp(substr(db2xml.generate_unique
(),1,8))),16)as ship_id, date, mode,
part_key from ship_tab) s where o.order_key=1 and p.price>2000
and o.order_key=o.order_key and s.part_key ORDER BY order_key, part_key,
ship_id</SQL_stmt>
```

RDB node composition: restrictions

The following restrictions apply:

- The condition associated with any lower level RDB node in the DAD must compare against a literal.
- The condition associated with a root_node describes the relationship between the tables involved in the RDB node composition. For example, a primary foreign key relationship.
- Each equality in the condition associated with a top-level RDB_node specifies the join relationship between columns of two tables and is applied separately from the other equalities. In other words, all the predicates connected by AND do not apply simultaneously for a single join condition, thereby simulating an outer join during document composition. The parent-child relationship between each pair of tables is determined by their relative nesting in the DAD. For example:

  ```
  <condition>order_tab.order_key=part_tab.order_key AND
  part_tab.part_key=ship_tab.part_key</condition>
  ```

Parameter changes for stored procedures that return CLOBs

If you have CLOB files that are larger than 1 MB, XML Extender provides a command file to redefine the stored procedure parameter. Download the `crtgenxc.zip` file from the DB2 XML Extender Web site. This ZIP file contains the following programs:

crtgenxc.zos.jci and crtgenxc.zos.cmd

For use with XML Extender for OS/390 V7, APAR PQ58249 and later.

crtgenxc.db2

For use on XML Extender V7.2 Fixpak 5 for UNIX and Windows.

crtgenxc.iseries

For use with XML Extender V5R1 for iSeries

Using an extraction program, such as PKZIP, extract the file appropriate for your operating system. Extract the file to the following locations:

OS/390 and z/OS

No requirement where the file is located.

UNIX or Windows
: No requirement where the file is located.

iSeries
: Place the file as a member into a file. (For example, put the file into DXXSAMPLES/SQLSTMT).

To specify the CLOB length: Open the file in an editor and modify the *resultDoc* parameter, shown in the following example.

```
out    resultDoc     clob(clob_size),
```

Size recommendation: The size limit of the *resultDoc* parameter depends on your system setup, but be aware that the amount specified in this parameter is the amount allocated by JDBC, regardless of the size of the document. The size should accommodate your largest XML files, but should not exceed 1.5 gigabytes.

To run the command file: From the DB2 command line and directory where the file is located, enter:

- **For OS/390 or z/OS:**

  Modify the JCL or CMD file contents. Submit the JCL or run the CMD from the USS.
- **For UNIX or Windows:** From the DB2 command line and directory where the file is located, enter:
  ```
  db2   -tf  crtgenxc.db2
  ```
- **For iSeries:** From the OS/400 command line, enter:
  ```
  RUNSQLSTM SRCFILE(DXXSAMPLES/SQLSTMT) SRCMBR(CRTGENXC) NAMING(*SQL)
  ```

  Where *DXXSAMPLES/SQLSTMT* matches the Library and File names into which you downloaded the file.

New XML composition stored procedures for CLOBs

Two more composition stored procedures have been developed. These are:

- db2xml.dxxGenXMLCLOB
- db2xml.dxxRetrieveXMLCLOB

These new stored procedures are similar to db2xml.dxxGenXML and db2xml.dxxRetrieveXML except that the XML document is returned in a CLOB and does not require a result table.

By using these stored procedures, you no longer need temporary or permanent tables for composed documents. This simplifies programming, especially in a multi-user client environment, and also reduces the instruction path length and improves throughput.

db2xml.dxxGenXMLClob and db2xml.dxxRetrieveXMLClob have the following benefits:

- They can be used in DB2 Universal Database EEE.
- They are supported on Windows, UNIX, iSeries, and OS/390 and z/OS.

dxxGenXMLClob

Purpose: As input, dxxGenXMLClob takes a buffer containing the DAD. It constructs XML documents using data that is stored in the XML collection tables that are specified by the <Xcollection> in the DAD and returns the first and typically the only XML document generated into the *resultDoc* CLOB.

Format:

```
dxxGenXMLClob(CLOB(100k)                            DAD               /*input*/
              integer                               overrideType,     /*input*/
              varchar(varchar_value)                override,         /*input*/
              CLOB(2G)                              resultDoc,        /*output*/
              integer                               valid,            /*output*/
              integer                               numDocs,          /*output*/
              long                                  returnCode,       /*output*/
              varchar(1024)                         returnMsg),       /*output*/
```

| Where *varchar_value* is 32672 for Windows and UNIX, 16366 for iSeries, and 32767
| for OS/390 and z/OS.

Parameters:

Table 3. dxxGenXMLClob parameters

Parameter	Description	IN/OUT parameter
DAD	A CLOB containing the DAD file.	IN
overrideType	A flag to indicate the type of *override* parameter: **NO_OVERRIDE** No override. **SQL_OVERRIDE** Override by an SQL_stmt **XML_OVERRIDE** Override by an XPath-based condition.	IN
override	Overrides the condition in the DAD file. The input value is based on the *overrideType*. **NO_OVERRIDE** A NULL string. **SQL_OVERRIDE** A valid SQL statement. Using *overrideType* requires that SQL mapping be used in the DAD file. The input SQL statement overrides the SQL_stmt in the DAD file. **XML_OVERRIDE** A string that contains one or more expressions in double quotation marks separated by the word and. Using this *overrideType* requires that RDB_node mapping be used in the DAD file	IN
resultDoc	A CLOB that contains the composed XML document.	OUT
valid	valid is set as follows: • If VALIDATION=YES then valid=1 for successful validation or valid=0 for unsuccessful validation. • If VALIDATION=NO then valid=NULL.	OUT
numDocs	The number of XML documents that would be generated from the input data. **Note:** Currently only the first document is returned.	OUT
returnCode	The return code from the stored procedure.	OUT
returnMsg	The message text that is returned in case of error.	OUT

dxxRetrieveXMLClob

Purpose: dxxRetrieveXMLClob enables document composition from relational data. This stored procedure also serves as a means for retrieving decomposed XML documents.

The requirements for using dxxRetrieveXMLClob are the same as the requirements for dxxGenXMLClob. The only difference is that the DAD is not an input parameter for dxxRetrieveXMLClob, but it is the name of an enabled XML collection.

Format:

```
dxxRetreiveXMLClob(CLOB(100k)                   collectionname,     /*input*/
              integer                           overrideType,       /*input*/
              varchar(varchar_value)            override,           /*input*/
              CLOB(2G)                          resultDoc,          /*output*/
              integer                           valid,              /*output*/
              integer                           numDocs,            /*output*/
              long                              returnCode,         /*output*/
              varchar(1024)                     returnMsg),         /*output*/
```

| Where *varchar_value* is 32672 for Windows and UNIX, 16366 for iSeries, and 32767
| for OS/390 and z/OS.

Parameters:

Table 4. dxxRetrieveXMLClob parameters

Parameter	Description	IN/OUT parameter
collectionName	The name of an enabled XML collection.	IN
overrideType	A flag to indicate the type of *override* parameter: **NO_OVERRIDE** No override. **SQL_OVERRIDE** Override by an SQL_stmt **XML_OVERRIDE** Override by an XPath-based condition.	IN
override	Overrides the condition in the DAD file. The input value is based on the *overrideType*. **NO_OVERRIDE** A NULL string. **SQL_OVERRIDE** A valid SQL statement. Using this *overrideType* requires that SQL mapping be used in the DAD file. The input SQL statement overrides the SQL_stmt in the DAD file. **XML_OVERRIDE** A string that contains one or more expressions in double quotation marks separated by the word and. Using this *overrideType* requires that RDB_node mapping be used in the DAD file	IN
resultDoc	The maximum number of rows in the result table.	IN
valid	valid is set as follows: • If VALIDATION=YES then valid=1 for successful validation or valid=0 for unsuccessful validation. • If VALIDATION=NO then valid=NULL.	OUT
numDocs	The number of XML documents that would be generated from the input data. NOTE: currently only the first document is returned.	OUT
returnCode	The return code from the stored procedure.	OUT
returnMsg	The message text that is returned in case of error.	OUT

XML RDB node composition: Multiple overrides are now allowed

In the previous version of DB2 XML Extender. multiple overrides on the same path were not supported. Only the first override was taken and the rest were ignored. Currently, all overrides specified will be accepted.

Example 1: You can specify multiple XML overrides on the same location path to refine set conditions in your search. In the following example, we compose an XML document from the two tables using the test.dad file.

The following example shows you how to write multiple XML override code allowing you to constrain your search results.

Table 5. Department Table

Department Number	Department Name
10	Engineering
20	Operations
30	Marketing

Table 6. Employee Table

Employee Number	Department Number	Salary
123	10	$98,000.00
456	10	$87,000.00
111	20	$65,000.00
222	20	$71,000.00
333	20	$66,000.00
500	30	$55,000.00

The DAD file **test.dad** illustrated below contains a condition comparing the variable *deptno* with the value 10. To override this condition so that the search is expanded to greater than 10 and less than 30 you must set the *override* parameter when calling *dXXGenXML* as follows:

″/ABC.com/Department>10 AND /ABC.com/Department<30″

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "C:\dxx_xml\test\dtd\dad.dtd">
<DAD>
<dtdid>E:\dtd\lineItem.dtd</dtdid>
<validation>NO</validation>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "C:\dxx_sml\test\dtd\LineItem.dtd"</doctype>
<root_node>
<element_node name="ABC.com">
<RDB_node>
<table name="dept" key="deptno"/>
<table name="empl" key="emplno"/>
<condition>dept deptno=empl.deptno</condition>
</RDB_node>

   <element_node name="Department" multi_occurrence="YES">
   <text_node>
   <RDB_node>
    <table name="dept"/>
    <column name="deptno"/>
    <condition>deptno=10</condition>
   </RDB_node>
  </text_node>

<element_node name="Employees" multi_occurrence="YES">
 <text_node>
  <RDB_node>
   <table name="dept"/>
    <column name="deptnot"/>
     <condition>deptno=10</condition>
   </RDB_node>
  </text_node>

<element_node name="Employees" multi_occurence="YES">
    <element_node name="EmployeeNo">
       <text_node>
         <RDB_node>
             <table  name="empl"/>
              <column name="emplno"/>
               <condition>emplno<500</condition>
            </RDB_node>
          </text_node>
         </element_node>
      <element_node name="Salary">
         <text_node>
           <RDB_node>
              <table name="empl"/>
              <column name="salary"/>
              <condition>salary>5000.00</condition>
            </RDB_node>
          </text_node>
        </element_node>
    </element_node>
    </element_node>
    </element_node>
    </root_node>
    </Xcollection>
```

To compose an XML document without an override, enter **tests2x mydb test.dad result_tab** or you can invoke dxxGenXML without setting an override. This will generate a document similar to this:

```
<?xml version="1.0">
<!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd">
<ABC.com>
<Department>10
  <Employees>
   <EmployeeNo>123</EmployeeNO>
   <Salary>98,000.00</Salary>
   </Employees>
   <Employees>
    <EmployeeNo>456</EmployeeNo>
    <Salray>87,000.00</Salary>
   </Employees>
   </Department>
   </ABC.COM>
```

To override the DAD file you can invoke dxxGenXML as mentioned above, or you can run the test2x program with the specified conditions:

```
tests2x mydb test.dad result_tab -o 2 "/ABC.com/Department>10 AND /ABC.com/Department<30"
```

which generates the following document.

```
<?xml version="1.0">
<!DOCTYPE Order SYSTEM "C:\dxx_xml\test\dtd\LineItem.dtd">
<ABC.com>
 <Department>20
  <Employees>
   <EmployeeNo>111</EmployeeNo>
   <Salary>65,000.00</Salary>
  </Employees>
   <EmployeeNo>222</EmployeeNo>
    <Salary>71,000.00</Salary>
    </Employees>
   <Employees>
    <EmployeeNo>333</EmployeeNo>
     <Salary>66,000.00</Salary>
    </Employees>
   </Department>
   </ABC.com>
```

XML RDB node composition: orderBy implemented

The orderBy option—the order in which data is sorted—was previously not supported. You can now control the way the sibling elements are sorted by using the orderBy option. In the sample DAD called *orderBy.dad* below, orderBy is used to sort the contents of the output document by *location* in descending order and *itemno* in ascending order. The default order for *location* and *itemno* is ascending.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<Xcollection>
<prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Catalog SYSTEM "d:\dtd\test.dtd"</doctype>
<root_node>
<element_node name="Catalog">
    <RDB_node>
       <table name="stocks" orderBy="location desc, itemno asc"/>
    </RDB_node>
     <element_node name="Product" multi_occurrence="YES">
      <element_node name="ItemNo">
        <text_node>
          <RDB_node>
            <table name="stocks"/>
            <column name="itemno"/>
          </RDB_node>
        </text_node>
      </element_node>
         <element_node name="WarehouseLocation">
        <text_node>
          <RDB_node>
            <table name="stocks"/>
           < column name="location"/>
          </RDB_node>
        </text_node>
      </element_node>
    </element_node>
  </element_node>
</root_node>
</Xcollection>
</DAD>
```

By invoking dxxGenXML with the DAD illustrated above, the following document will be generated. Alternatively, you can use **tests2x mydb orderby.dad result_tab**. This will also generate the document.

```
!DOCTYPE Catalog SYSTEM "d:\dtd\test.dtd">
<Catalog>
  <Product>
<WarehouseLocation>Z</WarehouseLocation>
 < /Product>
  <Product>
<ItemNo>33</ItemNo>
    <WarehouseLocation>Y</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>77</ItemNo>
   <WarehouseLocation>Y</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>44</ItemNo>
    <WarehouseLocation>X</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>55/ItemNo>
    <WarehouseLocation>Q</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>66/ItemNo>
    <WarehouseLocation>Q</WarehouseLocation>
  </Product>
</Catalog>
```

By changing the orderBy specification to the following **orderBy=″location asc, itemno desc**, the document below will be generated.

```
<?xml version="1.0"?>
<!DOCTYPE Catalog SYSTEM "d:\dtd\test.dtd">
<Catalog>
  <Product>
    <ItemNo>66</ItemNo>
    <WarehouseLocation>Q</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>55</ItemNo>
    <WarehouseLocation>Q</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>44</ItemNo>
    <WarehouseLocation>X</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>77</ItemNo>
    <WarehouseLocation>Y</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>33</ItemNo>
    <WarehouseLocation>Y</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>22</ItemNo>
    <WarehouseLocation>Z</WarehouseLocation>
  </Product>
  <Product>
    <ItemNo>11</ItemNo>
    <WarehouseLocation>Z</WarehouseLocation>
  </Product>
</Catalog>
```

XML composition: Successful completion messages are returned

Complete messages are now returned for XML composition stored procedures. For example, DXXQ020I XML is successfully generated after an XML document is composed. Previously, messages were not being returned.

XML composition: Composition from rows that have null values is supported

You can now use columns that have null values to compose XML documents. Previously, using such columns to compose XML documents caused the XML Extender to fail.

Example: The following example illustrates how you can generate an XML document from a table*MyTable* which has a row containing a null value in column *Col1*. The dad used in the example is called *nullcol.dad.*

```
<?xml version="1.0"?>
 <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
 <DAD>
 <validation>NO validation>NO>
<Xcollection>
<SQL_stmt>SELECT 1 as X, Col1 FROM MyTable order by X, Col1</SQL_stmt>
<prolog>?xml version="1.0"?prolog>?xml version="1.0"?>
 <doctype>!DOCTYPE Order SYSTEM "e:\t3xml\x.dtd">
 <root_node>
  <element_node name="MyColumn">
<element_node name="Column1" multi_occurrence="YES">
       <text_node>
       <column name="Col1"/>
       </text_node>
     </element_node>
   </element_node>
 </root_node>
 </Xcollection>
 </DAD>
```

MyTable

Col 1
1
3
-

Run: **tests2x mydb nullcol.dad result_tab** or use dxxGenXML to produce the following document. Note that the third *Column1* element represents a null value.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "e:\t3xml\x.dtd">
<MyColumn>
  <Column1>1</Column1>
 <Column1>3</Column1>
 <Column1></Column1>
</MyColumn>
```

Case for column names

The upper and lowercase treatment has changed for the **result_colname** and the **valid_colname** column names. Formerly, these column names were required to be upper case. With this fix, they can also be lower or mixed case.

XML RDB node decomposition: Attribute node changes

For a subtree of the DAD with element_nodes and attribute_nodes that map to same table the following change has been made:

- Attribute nodes no longer have to be the first children of the lowest common ancestor of the element nodes that map to the same table.
- Attribute nodes can appear anywhere in subtree, as long as they are not involved in a join condition.

Condition statements in RDB node DADs

Line ending characters allowed in condition elements

This update documents a fix for a bug in which line ending characters in the condition element, <condition>, caused an error. Line ending characters are now allowed in conditions statements.

Increased flexibility in the order of the condition predicates for the top RDB node condition element

The restriction for the order of <condition> elements in the root node in the DAD has been removed. Prior to Fixpak 5, when decomposing, the condition required that all predicates for columns in the same table be grouped together. For example: the following <condition> would generate an internal error because the columns for tableA are mixed with the columns of tableB.

```
<condition>
           tableA.col1=tableB.col11 AND
           tableB.col33=tableC.col40 AND
           tableA.col2=tableB.col22 AND
           tableB.col44=tableC.col42
</condition>
```

Additionally, the predicates had to be listed in order of table hierarchy, so that if tableA is a higher level table than tableB, all predicates involving tableA's columns must go before those involving tableB's columns, when they do not include tableA's columns. In the previous example, the condition predicate would also generate an internal error because a lower level table was specified before the higher table.

In Fixpak 5, there is no restriction in the order of the condition predicates.

Chapter
Preparing to use the XML Extender: administration

Sections
Planning for XML collections->Mapping Schemes for XML collections->Requirements when using RDB node mapping

Update
There are no ordering restrictions on predicates of the root node condition.

Code pages:

Fix for # sign in code page 277

Checking for # sign in code page 277 failed and has been fixed.

Performance improvements

The following performance improvements have been made for composition and decomposition.

- The length of the override parameter has been increased from 1KB to 32KB for UNIX and Windows. On iSeries and zSeries it is 16KB.

The 1KB override imposed a restriction on the length of the SQL statement for SQL composition. The restriction encouraged the use of database views to reduce the length of the required SQL statement. However, that database views can sometimes incur additional path length because of view materialization. With a long override, the strong need for views is reduced. Note that this override parameter does not apply to the MQSeries stored procedures. The override for those stored procedures is still 1KB, for example. VARCHAR(1024).

- The requirement for an intermediate result table has been removed.

By using these stored procedures:

- You reduce the instruction path length because there is no need to create result tables.
- You simplify your programming.

Use the stored procedures that require an intermediate result table if you want to produce more than one document.

- The user defined functions for XML column have been enhanced for performance

The DB2 XML Extender user-defined functions will now keep small (512KB) XML documents in memory while processing them. This reduces Input/Output activity and the contention for the disk that is used for temporary files.

The definition of the DB2 XML Extender scalar (non-table) user-defined functions (UDFs) has been changed so that they can be run in parallel. This provides significant performance improvements in the execution of queries that refer to the UDFs more than once. You have to run the migration script program to get the parallel capability for the scalar UDFs. If you already have columns enabled using the scalar UDFs, disable all XML columns, run the migration script and then re-enable the columns.

Chapter 3. Message updates

The following sections describe new and changed messages for the XML Extender

Updated error messages

The following messages had missing list items in the previous versions of *DB2 Universal Database XML Extender Administration and Programming, Version 7* . The full messages are listed in the following sections.

Error message DXXA027E

Message
Could not disable the column.

Explanation

XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User response
Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA028E

Message
Could not disable the column.

Explanation

XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User response
Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA029E

Message
Could not disable the column.

Explanation

XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User response
Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA030E

Message
Could not disable the column.

Explanation

XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User response
Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA047E

Message
Could not enable the column.

Explanation

XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA048E

Message
Could not enable the column.

Explanation

XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA049E

Message
Could not enable the column.

Explanation

XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA050E

Message
Could not enable the column.

Explanation

XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA051E

Message
Could not disable the column.

Explanation

XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User response
Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA052E

Message
Could not disable the column.

Explanation

XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA053E

Message

Could not enable the column.

Explanation

XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

Error message DXXA054E

Message

Could not enable the column.

Explanation

XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User response

Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

New error messages

The following messages have been added to the XML Extender.

Error message DXXC013E

Message

The results of the extract UDF exceed the size limit for the UDF return type.

Explanation

The data returned by an extract UDF must fit into the size limit of the return type of the UDF, as defined in the DB2 XML Extenders Administration and Programming guide. For example, the results of extractVarchar must be no more than 4000 bytes (including the terminating NULL).

User response

Use an extract UDF that has a larger size limit for the return type: 254 bytes for extractChar(), 4 KB for extractVarchar(), and 2 GB for extractClob().

Error message DXXQ038E

Message

The SQL statement is too long: *SQL_statement*.

Explanation

The SQL statement specified in SQL_stmt element of DAD exceeds the allowed number of bytes.

User response

Reduce the length of SQL statement to less than or equal to 32765 bytes for Windows and UNIX, or 16380 bytes for OS/390 and OS/400.

Error message DXXQ039E

Message

Too many columns specified for a table in the DAD file.

Explanation

A DAD file used for decomposition or RDB composition can have a total of at most 100 text_node and attribute_node elements that specify unique columns within the same table. For example, the following fragment of a DAD contains two text_node elements that specify two different columns within the same table SHIP_TAB:

```
<element_node name="ShipMode">
    <text_node>
       <RDB_node>
           <table name="ship_tab"/>
           <column name="mode"/>
       </RDB_node>
     </text_node>
</element_node>
<element_node name="Comment">
     <text_node>
       <RDB_node>
           <table name="ship_tab"/>
           <column name="comment"/>
       </RDB_node>
     </text_node>
</element_node>
```

User response

Reduce the total number of text_node and attribute_node elements that refer to unique columns within the same table to no more than 100.

Error message DXXQ040E

Message

The element name *element_name* in the DAD file is invalid.

Explanation

The specified element name in document access definition (DAD) file is wrong.

User response
Ensure that the element name is typed correctly in the DAD file. See the DTD for the DAD file.

Error message DXXQ041W

Message
XML document successfully generated. One or more override paths specified is invalid and ignored.

Explanation
One or more override path specified is invalid and ignored. This warning message is issued when a stored procedure has been run with an XML override. More that one path was specified, and only the first one was used. Multiple override paths are not supported.

User response
Specify only override path.

Error message DXXQ043E

Message
Attribute *attr_name* not found under element *elem_name*.

Explanation
The attribute *attr_name* was not present in element *elem_name* or one of its child elements.

User response
Ensure the attribute appears in the XML document everywhere that the DAD requires it.

Error message DXXQ044E

Message
Element *elem_name* does not have an ancestor element *ancestor*.

Explanation
According to the DAD *ancestor* is an ancestor element of *elem_name*. In the XML document, one or more element *elem_name* does not have such an ancestor.

User response
Ensure the nesting of elements in the XML document conforms to what is specified in the corresponding DAD.

Error message DXXQ045E

Message
Subtree under element *elem_name* contain multiple attributes named *attrib_name*.

Explanation
A subtree under *elem_name* in the XML document contains multiple instances of attribute *attrib_name*, which according to the DAD, is to be decomposed into the same row. Elements or attributes that are to be decomposed must have unique names.

User response
Ensure that the element or attribute in the subtree has a unique name.

Trademarks

Printed in U.S.A.