August 2007

DB2 Information Management Software

Guide to conversion User Defined Functions (UDFs)

By Preethi Vishwanath , Randall P. Spalten
Software Engineer Advisory Programmer
IBM Software Group

1. Overview

There are different alternatives available to insert documents that contain characters which cannot be represented in the database codepage. One of the approaches would be to convert the problem characters into hexadecimal character references (of the form "&#xhhhh;", where hhhh is the hexadecimal Unicode UTF16 code point of the character). Decimal character references can be used in any XML fragment and is replaced by the actual code point during XML parsing. The character string "I just joined the ΔΨ΀ fraternity!" is equivalent to "I just joined the ΔΨΠ fraternity!" in UTF-8.

To assist the user in converting XML documents, DB2 provides 2 UDFs that test and clean up XML documents before they are inserted into the database. This document would serve as a user manual for these User Defined Functions.

2. User Defined Functions

There are 2 different UDFs which have been provided as a part of this deliverable.

TEST_XML
CLEAN_XML

a. TEST_XML

TEST_XML takes in a Binary Large Object “BLOB” (the preferred method is to use a BLOB_FILE referencing an XML text file encoded in UTF-8 code set) containing the XML document, and outputs a Boolean value. When TEST_XML is called, DB2 attempts to convert the BLOB from UTF-8 into the database code page, and return TRUE if no substitution characters were encountered or FALSE if some substitution character was encountered during the conversion. This does not insert the document or modify the BLOB in any way. It is simply a test to see whether this XML document can be safely inserted as a CHAR, VARCHAR, or Character Large Object “CLOB” into the database without resulting in data integrity loss.

Input: BLOB , Integer <Optional>
Output: Integer
Function Declaration:

```
TEST_XML (input BLOB(2G)) RETURNS INTEGER
```

If the output observed is a “1”, it means there was some substitution character detected, i.e.; users cannot insert this document into the database, without using CLEAN_XML or, maybe by passing this document in as a BLOB etc. Insertion of unsafe documents to the database could result in lose of data.

If the output observed is a “0”, it means that no substitution character was observed and the document could be inserted safely into the database.

If the BLOB is encoded in another code page other than UTF-8, users can specify the code page of the BLOB input as an optional second parameter to TEST_XML.

```
TEST_XML (input BLOB(2G),INTEGER) RETURNS INTEGER
```

b. CLEAN_XML

CLEAN_XML performs conversion with substitution escape character replacement. It takes in a BLOB or BLOB file containing an XML document (assumed to be in UTF-8 code page), and outputs a CLOB containing the XML document in the database code page, with each code point that cannot be converted safely into the database code page replaced with the escape character form "&#xhhhh", where hhhh is the hexadecimal UTF16 code point of the character.

This does not insert the document or modify the BLOB in any way, but it can be used in conjunction with INSERT/XMLPARSE to safely insert any given BLOB XML document into any database, for example:

```
INSERT INTO XTAB VALUES (XMLPARSE(CLEAN_XML(:BLOB_HV)));
```

If the BLOB is encoded in another code page other than UTF-8, users can specify the code page of the BLOB input as an optional second parameter to CLEAN_XML.

As the name suggests this UDF cleans the XML document by replacing the characters which contain code points not present in the database codepage by their hexadecimal equivalent.

Input: BLOB , Integer <Optional>
Output: CLOB (codepage of the database)

Function Declaration:

```
CLEAN_XML (input BLOB(2G)) RETURNS CLOB(2G)
```

Sample Input on database with code set ISO-8859-7:

```
<?xml version="1.0" encoding="utf-8" ?>
  <product pid="100-101-10">
   <description>
     <name>Stérlíng Sílvér Sugar Créamér by Pöölé</name>
     <details>Spéctacular sugar Pöölé "Öld English"
</details>
     <price currency="és" alias="é">35</price>
  </description>
</product>
```

Sample Output on database with code set ISO-8859-7:

```
<?xml version="1.0" encoding="utf-8" ?>
  <product pid="100-101-10">
   <description>
```

```
        <name>St&#xE9;rl&#xED;ng S&#xED;lv&#xE9;r Sugar
Cr&#xE9;am&#xE9;r by P&#xF6;&#xF6;l&#xE9;</name>
        <details>Sp&#xE9;ctacular sugar
P&#xF6;&#xF6;l&#xE9; "&#xD6;ld Engl&#xED;sh" </details>
        <price currency="&#xE9;s"
alias="&#xE9;">35</price>
  </description>
</product>
```

If the BLOB is encoded in another code page other than UTF-8, users can specify the code page of the BLOB input as an optional second parameter to CLEAN_XML.

```
CLEAN_XML (input BLOB(2G),Integer) RETURNS CLOB(2G)
```

3. Description

The zipped folder (attached with this document) consists of a set of scripts which have been brought together to provide the user with the UDFs. We have also provided a small sample embedded C client program which invokes the UDFs. Please note that our conversion process makes use of certain ICU (http://www.icu-project.org/) libraries and header files.

Platform tested: AIX
Sample database: sample

testconv.c

testconv (C program) defines the User Defined Functions. Function substitute_subchar_udf corresponds to CLEAN_XML while test_subchar_udf corresponds to TEST_XML.

testconv.db2

testconv.db2 serves as the DB2 counterpart for the User Defined Functions (TEST_XML,CLEAN_XML).
Execution: db2 –tvf testconv.db2

testconvcli_1.sqc

testconvcli_1.sqc serves as a sample client program which invokes the User Defined Functions TEST_XML,CLEAN_XML (with the optional codepage parameter absent).
Execution: testconvcli_1 [database]

testconvcli_2.sqc

testconvcli_2.sqc serves as a sample client program which invokes the User Defined Functions TEST_XML,CLEAN_XML (with optional codepage parameter being passed as an argument).
Execution: testconvcli_2 [database]

run_1.sh:

run_1.sh acts as an automation script which performs for the client-server execution of UDFs TEST_XML and CLEAN_XML (with the optional codepage parameter absent)

i. Setting up the environment (by default database with code set ISO-8859-1).
ii. Building the source and client programs.
iii. Executing the source and client programs.

run_2.sh:

run_2.sh acts as an automation script which performs for the client-server execution of UDFs TEST_XML and CLEAN_XML (with optional codepage parameter being passed as an argument).

i. Setting up the environment (by default database with code set ISO-8859-1).
ii. Building the source and client programs.
iii. Executing the source and client programs.

Makefile:

This file serves as a makefile for our samples on AIX.

Execution options:
make all: Builds both the server and the client program.
make testconv: To build only the User Defined Functions
make testconvcli_1: To build only the client program to invoke TEST_XML and CLEAN_XML (without optional second parameter).
make testconvcli_2: To build only the client program to invoke TEST_XML and CLEAN_XML (with codepage argument passed).
make clean: Cleans all the object files, bind files and the executables.

bldrtn:

Builds AIX C routines (stored procedures and UDFs)
bldrtn <prog_name> [<db_name>]

bldapp:

Builds AIX C application programs.
bldapp <prog_name> [<db_name> [<userid> <password>]]

setup.db2:

Basic table structure setup.

Execution:
db2 –tvf setup.db2

test.data:

test.data would be the XML document which the user needs to verify or clean before inserting to the database. This file serves as the input file to be provided to our client sample program.

There are three different alternatives available for the users to be able to provide their document as input.

i. Overwrite the XML of test.data with the users XML document
ii. Rename users document as test.data
iii. Modify the filename parameter in testconvcli_1.sqc or testconvcli_2.sqc accordingly to user's filename.

Please note the sample provided contains symbols which belong to ISO-8859-1 client code page.

4. Execution Steps

For ease to use, these commands have been put together into an automated script “run_1.sh” and “run_2.sh”

Sample Script:

```
# Default codepage 819 , ISO-8859-1
# If any other codepage ,Change DB2CODEPAGE parameter
db2set DB2CODEPAGE=819
db2stop force
db2start
# Default database name: sample
# If any other database, replace sample with new db name .
make clean
db2 -tvf setup.db2
make testconv
db2 -tvf testconv.db2
make testconvcli_1
# Default database name : sample
# If any other database, replace sample with new db name .
testconvcli_1 sample
```

5. Error Handling

Scenario 1:

Incorrect File name:

Error Observed:

```
---- error report ----------------------------

application message = BLOB data -- write
line              = 100
file              = testconvcli.sqc
SQLCODE           = -452

SQL0452N  Unable to access the file referenced by host variable "1".
Reason
code: "3".  SQLSTATE=428A1

SQLSTATE 428A1: Unable to access a file referenced by a host file
variable.

---- end error report ------------------------
```

User Response:

Check the name of the input data file being passed in testconvcli_1.sqc / testconvcli_2.sqc and modify accordingly.

Scenario 2:

Incorrect Code Page:

Error Observed:
SQL0443N Routine "TEST_XML" (specific name "") has returned an error SQLSTATE
with diagnostic text " InCorrect code page ….". SQLSTATE=38100

User Response:

Check code page provided. For a list of permissible code pages please refer

Scenario 3:

ICU related errors

Error Observed:
SQL0443N Routine "TEST_XML" (specific name "") has returned an error SQLSTATE
with diagnostic text " ICU Unicode Converter Open Failed ".
SQLSTATE=38100

User Response:

Check whether the ICU path provided is correct.

#. Related Readings (optional)

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0707spalten/#N101B5

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada