Db2 for z/OS:

SQL Table UDF returns result set of Stored Procedure

[via array, XML, JSON, or REST]

Example: Display zParms in SPUFI

© Walter E. Huth, 2021-2024

Motivation / Background / Goal / … [1]

- **Positive:** There are many Stored Procedures
  (self-coded and Db2-provided,
  e.g. ADMIN_INFO_SYSPARM() lists zParms)
- **Negative:** Can‘t use stored procedure in SPUFI, DSNTEP2/4
- **Goal: SQL Table UDF returning result set of stored procedure**

  (why **SQL** Table UDF ? – easier!)
- **Solution alternatives** follow ...

© Walter E. Huth, 2021-2024

Motivation / Background / Goal / … [2]

- **Problem of SQL Table UDF: Only one fullselect!**
  → definition must not include SQL PL, especially: no CALL sp

- **Goal: Create SQL Table UDF containing SQL PL, CALL sp**

(why **SQL** Table UDF ? – easier!)

Solution alternatives follow ...

© Walter E. Huth, 2021-2024

Caution

- **Warning: only „proof of concept“**
  - you may enhance the code, e.g.,
    - extensive error handling
    - performance improvements
    - test in a Data Sharing environment
    - explore max. amount of data

- **In DB2 LUW,**
  there may exist an alternative to the solutions presented here
  - The idea is that a pipelined table function can consume the result-set and pipe it to the caller.
  - Not applicable to DPF, DB2 i, nor DB2 z
  - see https://stackoverflow.com/questions/57601980/trying-to-run-dynamic-sql-using-a-udf-in-db2

© Walter E. Huth, 2021-2024

Agenda

- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet

  ← Example: Get one zParm value

- **SQL Table UDF „zParms . ()“**

  ← Example: List all zParm values

  - invokes SQL scalar UDF „zParms._helper()“
    that calls Stored Procedure [*]

    Data transfer from scalar UDF to table UDF via:
    - Temporary Table: „zParms**d/c**()“
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“

    ← **d**eclared / **c**reated: both do not work

  Alternative !

  - invokes RESTful service
    that calls Stored Procedure [*]

    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParm_value() (1/4)

```
-- SQL UDF "zParm_value()" that calls admin_info_sysparm
--   input:  zParm name
--   output: zParm value
--   assumption: non data sharing system
--
CREATE FUNCTION zParm_value ( i_zParm_name VARCHAR(40) )
   RETURNS VARCHAR(2048)
   LANGUAGE SQL
   MODIFIES SQL DATA           -- required, as calling admin_info_sysparm
   BEGIN
      DECLARE SQLCODE          INTEGER DEFAULT 0 ;
      DECLARE SQLSTATE         CHAR(5) DEFAULT '00000' ;      -- not used
      -- output string:
      DECLARE o_string         VARCHAR(2048) ;
...
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParm_value() (2/4)

```
-- input parameter for admin_info_sysparm:
DECLARE db2_member            CHAR(8) ;             -- here: NULL
DECLARE retcd                 INTEGER ;
DECLARE err_msg               VARCHAR(1331) ;
-- result set locator:

DECLARE rs_loc1               RESULT_SET_LOCATOR VARYING ;

-- result set row from admin_info_sysparm:
DECLARE rownum                INTEGER ;
DECLARE macro                 CHAR(9) ;             -- not used
DECLARE parameter             VARCHAR(40) ;
DECLARE install_panel         VARCHAR(8) ;          -- not used
DECLARE install_field         VARCHAR(40) ;         -- not used
DECLARE install_location      VARCHAR(12) ;         -- not used
DECLARE value                 VARCHAR(2048) ;
DECLARE additional_info       VARCHAR(200) ;        -- not used
-- other variables:
DECLARE at_end                INTEGER ;
DECLARE start_timestamp       TIMESTAMP ;
DECLARE duration              DECIMAL(20,6) ;
--
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParm_value() (3/4)

```
CALL sysproc.admin_info_sysparm
     ( NULL , retcd , err_msg ) ;    -- NULL: non data sharing
IF sqlcode = 466 THEN                -- resultSet returned
     SET start_timestamp = current_timestamp ;
     ASSOCIATE LOCATORS (rs_loc1 )
               WITH PROCEDURE sysproc.admin_info_sysparm ;
     ALLOCATE c1 CURSOR FOR RESULT SET rs_loc1 ;
     FETCH c1 INTO rownum            -- a number
                 , macro
                 , parameter         -- zParm name
                 , install_panel
                 , install_field
                 , install_location
                 , value             -- zParm value
                 , additional_info ;
     --
     SET i_zParm_name = UPPER ( i_zParm_name ) ;
     SET at_end = 0 ;
     IF ( i_zParm_name = parameter ) THEN
       SET o_string = value ;
       SET at_end = 1 ;
     END IF ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParm_value() (4/4)

```
      WHILE at_end = 0 DO
         FETCH c1 INTO rownum, macro, parameter
                     , install_panel, install_field
                     , install_location, value, additional_info ;
         IF (i_zParm_name = parameter) THEN
            SET o_string = value ;
            SET at_end = 1 ;
         END IF ;
         IF ( sqlcode = 100 ) THEN SET at_end = 1 ;
         END IF ;
         END WHILE ;
      CLOSE  c1 ;
      SET duration = current_timestamp - start_timestamp ;
      SET o_string = i_zParm_name CONCAT ' : ' CONCAT o_string
             CONCAT '     (Duration: '
             CONCAT duration CONCAT ' seconds )' ;
      RETURN o_string ;
    ELSE RETURN 'no  +466 (result set)  returned' ;
    END IF ;
 END  -- from begin
#
```

© Walter E. Huth, 2021-2024

Run SQL scalar UDF zParm_value()

```
SELECT zParm_VALUE ('FCCOPYDDN')                AS "zParm : WERT"

  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT zParm_VALUE ('FLASHCOPY_LOAD')           AS "zParm : WERT"

  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT zParm_VALUE ('FLASHCOPY_REBUILD_INDEX')  AS "zParm : WERT"

  FROM SYSIBM.SYSDUMMY1
UNION ALL
SELECT zParm_VALUE ('FLASHCOPY_REORG_INDEX')    AS "zParm : WERT"

  FROM SYSIBM.SYSDUMMY1
```

```
  zParm : WERT
  ---------+---------+---------+---------+---------+---------+---------+
  FCCOPYDDN : HLQ.&DB..&SN..N&DSNUM..&UQ.      (Duration: .058121 seconds )
  FLASHCOPY_LOAD : NO      (Duration: .059130 seconds )
  FLASHCOPY_REBUILD_INDEX : NO      (Duration: .058484 seconds )
  FLASHCOPY_REORG_INDEX : NO      (Duration: .059980 seconds )
  DSNE610I NUMBER OF ROWS DISPLAYED IS 4
  DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
```

© Walter E. Huth, 2021-2024

Remark: Why SQL UDF?

No hassle with

- Programming details
  (Indicator variables, SQL delimiter)
- Program preparation steps
  (Precompile, Compile, Link, BIND PACKAGE)
- Administration of artefacts
  (LOAD module, DBRM, Package)
- WLM application environment
  (definition, start procedure, QUIESCE/RESUME/REFRESH)

© Walter E. Huth, 2021-2024

Agenda

- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet

  Example:
  Get one zParm value

- **SQL Table UDF „zParms . ()“**
  - invokes SQL scalar UDF „zParms ._helper()“
    that calls Stored Procedure [*]

    Example:
    List all
    zParm values

    Data transfer from scalar UDF to table UDF via:
    - Temporary **T**able: „zParms**d/c**()“ ← does not work
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“

  Alternative !

  - invokes RESTful service
    that calls Stored Procedure [*]

    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

© Walter E. Huth, 2021-2024

Challenge of SQL Table UDF: only one Fullselect

- **In other words: No SQL PL**
- Bypass the restriction:
  - in a CTE:
    - invoke another („helper“) UDF, a scalar UDF
    - **which contains SQL PL**
    - and returns the intended output
  - in the main SELECT:
    - process the scalar UDF‘s result to produce a nice output

```
WITH tmp-tab (col-list) AS
( SELECT scalar-helper-udf() )
SELECT ...
  FROM tmp-tab
```

Fullselect of the SQL Table UDF

© Walter E. Huth, 2021-2024

Goal: Using SQL table UDF “zParms()”

```
SELECT rownum                          -- or:  SELECT zp.*
     , parameter
     , value

 FROM TABLE (zParms_('')) zp           -- optional input: Db2 member name

 WHERE parameter LIKE '%COPY%D%'
 ORDER BY parameter
#
```

```
---------+---------+---------+---------+---------+---------+---------+---------+--
     ROWNUM  PARAMETER                                VALUE
---------+---------+---------+---------+---------+---------+---------+---------+--
        208  FCCOPYDDN                                HLQ.&DB..&SN..N&DSNUM..&UQ.
        210  FLASHCOPY_LOAD                           NO
        213  FLASHCOPY_REBUILD_INDEX                  NO
        214  FLASHCOPY_REORG_INDEX                    NO
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+--
```

© Walter E. Huth, 2021-2024

Agenda

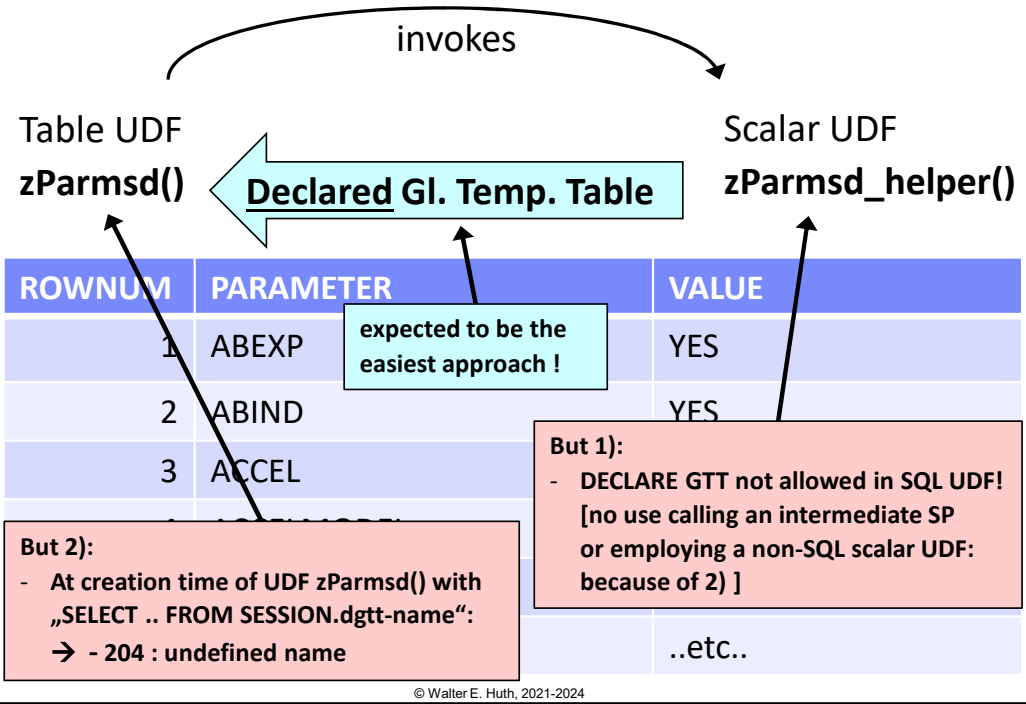
- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet
  ← Example: Get one zParm value
- **SQL Table UDF „zParms . ()“**
  ← Example: List all zParm values
  - invokes SQL scalar UDF „zParms ._helper()“
    that calls Stored Procedure [*]
    Data transfer from scalar UDF to table UDF via:
    - Temporary **T**able: „zParms**d/c**()“ ← does not work
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“
  - invokes RESTful service
    that calls Stored Procedure [*]
    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

Alternative !

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

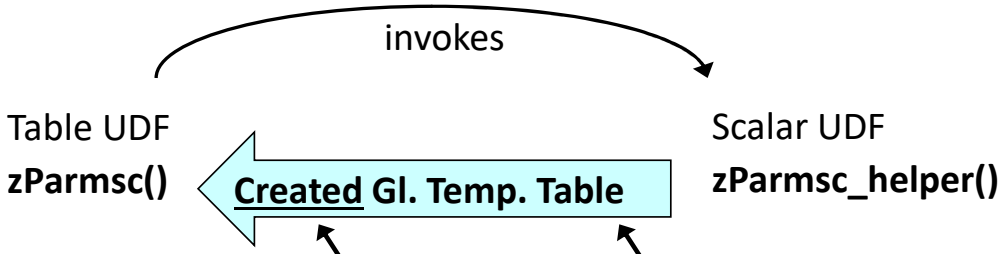
© Walter E. Huth, 2021-2024

Table UDF gets data via declared temp. table



© Walter E. Huth, 2021-2024

Table UDF gets data via created temp. table



ROWNUM	PARAMETER	VALUE
0	Duration or zParmsc_helper()	00 min 00.254196 sec
1	ABEXP	YES
2	ABIND	YES
3	ACCEL	NO
4	ACCELMODEL	YES
..etc..	..etc..	..etc..

That should work, as "name defined"!

Table definition required !

© Walter E. Huth, 2021-2024

Create CGTT "zParms_tab"

```
CREATE GLOBAL TEMPORARY TABLE zParms_tab
       ( rownum     INTEGER
       , parameter  VARCHAR(  40)
       , value      VARCHAR(2048)
       )
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsc_helper() (1/5)

```
-- SQL scalar udf zParmsc_helper() that supports table udf zParmsc()
-- by inserting rows into created global temporary table zParms_tab
--   input parameter: db2_member_name (if non data sharing: '' or ' ')
--   calls: sysproc.admin_info_sysparm
--   output: #rows or error message (only used for test purposes)
--
CREATE FUNCTION zParmsc_helper ( i_db2_member VARCHAR(8) )
  RETURNS VARCHAR(200)

  LANGUAGE SQL
  MODIFIES SQL DATA   -- required, as udf calls admin_info_sysparm
                      -- else sqlcode -577
  BEGIN
    DECLARE SQLCODE            INTEGER DEFAULT 0 ;
    DECLARE SQLSTATE           CHAR(5) DEFAULT '00000' ;    -- not used
    -- output value:
    DECLARE o_string              VARCHAR(200)   ;
    -- parameters for admin_info_sysparm:
    DECLARE db2_member            CHAR(8) ;
    DECLARE retcd                 INTEGER ;
    DECLARE err_msg               VARCHAR(1331) ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsc_helper() (2/5)

```
    -- result set locator:
    DECLARE rs_loc1               RESULT_SET_LOCATOR VARYING ;

    -- result set row:
    DECLARE rownum                INTEGER ;
    DECLARE macro                 CHAR(9) ;          -- not used
    DECLARE parameter             VARCHAR(40) ;
    DECLARE install_panel         VARCHAR(8) ;       -- not used
    DECLARE install_field         VARCHAR(40) ;      -- not used
    DECLARE install_location      VARCHAR(12) ;      -- not used
    DECLARE value                 VARCHAR(2048) ;
    DECLARE additional_info       VARCHAR(200) ;     -- not used
    -- other variables:
    DECLARE rownum_char           CHAR(10) ;
    DECLARE at_end                SMALLINT ;
    DECLARE start_timestamp       TIMESTAMP ;      -- optional
    DECLARE duration              DECIMAL(20,6) ;  -- optional
    DECLARE duration_char         VARCHAR(100) ;   -- optional
    DECLARE number_of_rows        VARCHAR( 10) ;   -- optional
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsc_helper() (3/5)

```
SET start_timestamp = CURRENT_TIMESTAMP ;

CALL SYSPROC.ADMIN_INFO_SYSPARM
     ( db2_member , retcd , err_msg ) ;

IF SQLCODE = 466 THEN

 ASSOCIATE LOCATORS (rs_loc1 )
           WITH PROCEDURE SYSPROC.ADMIN_INFO_SYSPARM ;
 ALLOCATE c1 CURSOR FOR RESULT SET rs_loc1 ;

  FETCH c1 INTO rownum, macro, parameter
      , install_panel , install_field  , install_location
      , value  , additional_info  ;
  INSERT INTO zParms_tab VALUES
      ( rownum, parameter, value ) ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsc_helper() (4/5)

```
SET at_end = 0 ;
WHILE at_end = 0 DO
  FETCH c1 INTO rownum, macro, parameter
      , install_panel, install_field, install_location
      , value, additional_info    ;
  IF ( SQLCODE = 100 ) THEN SET at_end = 1 ;
  END IF ;
  INSERT INTO zParms_tab VALUES
    ( rownum, parameter, value ) ;
  END WHILE ;
CLOSE  c1 ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsc_helper() (5/5)

```
       -- add additional duration-info:
       SET duration = CURRENT_TIMESTAMP - start_timestamp ;
       SET rownum = 0 ;
       SET duration_char = SUBSTR(DIGITS(duration),11,2)
                           CONCAT ' min ' CONCAT
                           SUBSTR(DIGITS(duration),13,2)
                           CONCAT '.'     CONCAT
                           SUBSTR(DIGITS(duration),15,6)
                           CONCAT ' sec' ;
       INSERT INTO zParms_tab VALUES
           ( rownum, 'Duration of zParmsc_helper: ', duration_char ) ;
       -- end of additional duration info --
       SET number_of_rows
           = ( SELECT DIGITS(COUNT(*)) FROM zParms_tab ) ;
       SET o_string = '#rows = ' CONCAT number_of_rows ;
     ELSE SET o_string = 'NO  +466 (RESULT SET)  returned' ;
     END IF ;
     RETURN o_string ;
   END  -- of BEGIN
#
```

© Walter E. Huth, 2021-2024

Create SQL Table UDF “zParmsc()”

```
CREATE FUNCTION zParmsc ( i_db2_member VARCHAR(8) )
   RETURNS TABLE ( rownum     INTEGER
                 , parameter  VARCHAR(  40)
                 , value      VARCHAR(2048)
                 )
   LANGUAGE SQL
   RETURN
    WITH temp ( out_col ) AS

    ( SELECT zParmsc_helper('') AS out_col

        FROM sysibm.sysdummy1
    )
    SELECT rownum, parameter, value
      FROM zParms_tab
#
```

1. **zParmsc_helper()** inserts into CGTT zParms_tab

2. Reading CGTT zParms_tab

© Walter E. Huth, 2021-2024

Select from SQL Table UDF “zParmsc()”

```
SELECT zp.*
  FROM TABLE (zParmsc('')) zp       -- optional input: Db2 member name
#
```

```
---------+---------+---------+---------+---------+---------+---------+
     ROWNUM  PARAMETER                                   VALUE
---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
```

?

© Walter E. Huth, 2021-2024

Inquiry (1)

```
select zParmsc_helper('')
  from sysibm.sysdummy1
#
---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+
#rows = 0000000355
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
WITH temp (out_col) AS
( SELECT zParmsc_helper('')  AS out_col
    FROM sysibm.sysdummy1
)
SELECT *
  FROM zParms_tab
#
---------+---------+---------+---------+---------+---------+---------+
     ROWNUM  PARAMETER                                   VALUE
---------+---------+---------+---------+---------+---------+---------+
         1  AUDITST                                     000000000000000..
         2  CONDBAT                                     0000001000
         3  CTHREAD                                     00200
         4  DLDFREQ                                     ON
         5  PCLOSEN                                     00010
```

1. Inserts into CGTT zParms_tab

2. Query within same process

!

© Walter E. Huth, 2021-2024

Inquiry (2)

```
select zParmsc_helper('')
  from sysibm.sysdummy1
 #
---------+---------+---------+---------+---------+---------+---------+

---------+---------+---------+---------+---------+---------+---------+
#rows = 0000000355
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
```

NO previous inserts into CGTT

```
WITH temp (out_col) AS
( SELECT zParmsc_helper('') AS out_col
    FROM sysibm.sysdummy1
)
SELECT *
  FROM zParms_tab
 #
---------+---------+---------+---------+---------+---------+---------+
     ROWNUM  PARAMETER                                        VALUE
---------+---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------
```

Query with same CTE does **not** insert into CGTT zParms_tab !

© Walter E. Huth, 2021-2024

Inquiry (3)

All my trials
to **force Db2 to insert into CGTT „zParms_tab“**
[that is, to execute zParmsc_helper()]
before selecting from it (within one query)
were **unsuccessful**, e.g.:

Ideas are welcome!

```
WITH temp (out_col) AS
( SELECT zParmsc_helper('')
    FROM sysibm.sysdummy1
  UNION  ALL
  SELECT 'PCLOSEN'
    FROM sysibm.sysdummy1
)
SELECT *
  FROM temp, zParms_tab
#
---------+---------+---------+---------+---------+---------+
OUT_COL
---------+---------+---------+---------+---------+---------+
DSNE610I NUMBER OF ROWS DISPLAYED IS 0
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+
```

© Walter E. Huth, 2021-2024

Agenda

- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet
- **SQL Table UDF „zParms . ()“**
  - invokes SQL scalar UDF „zParms ._helper()“
    that calls Stored Procedure [*]
    Data transfer from scalar UDF to table UDF via:
    - Temporary **T**able: „zParms**d/c**()“
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“
  - invokes RESTful service
    that calls Stored Procedure [*]
    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

Example: Get one zParm value

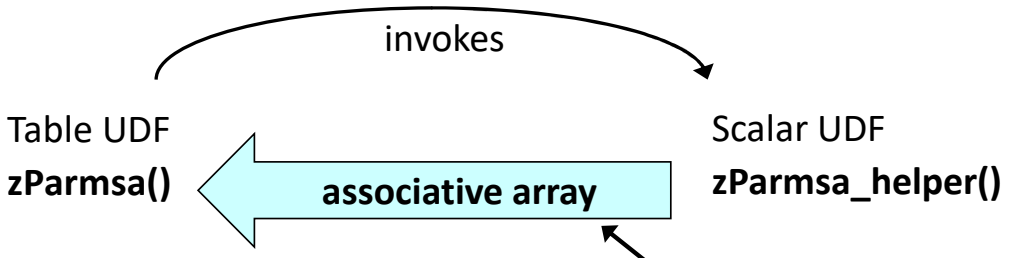
Example: List all zParm values

does not work

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

© Walter E. Huth, 2021-2024

zParmsa() gets data via associative array



PARAMETER (= array index)	VALUE
*** Duration of helper UDF: ***	00 min 00.254196 sec
ABEXP	YES
ABIND	YES
ACCEL	NO
ACCELMODEL	YES
..etc..	..etc..

Array definition required !

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsa_helper() (1/5)

```
-- SQL scalar udf zParmsj_helper() that supports table udf zParmsj()
--   input parameter: db2_member_name (if non data sharing: '' or ' ')
--   calls: sysproc.admin_info_sysparm
--   output: associative array
--
-- create associative array for zParm names and their values:
CREATE TYPE ass_array_4zParms AS VARCHAR(2048) ARRAY [VARCHAR(40)]
#
CREATE FUNCTION zParmsa_helper ( i_db2_member VARCHAR(8) )
  RETURNS ass_array_4zParms
  LANGUAGE SQL
  MODIFIES SQL DATA            -- as call of admin_info_sysparm , else -577
  BEGIN
    DECLARE SQLCODE            INTEGER DEFAULT 0 ;
    DECLARE SQLSTATE           CHAR(5) DEFAULT '00000' ;    -- not used
    -- output value:
    DECLARE zParms_array_var   ass_array_4zParms ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsa_helper() (2/5)

```
    -- parameters for admin_info_sysparm:
    DECLARE db2_member           CHAR(8) ;
    DECLARE retcd                INTEGER ;
    DECLARE err_msg              VARCHAR(1331) ;
    -- result set locator:
    DECLARE rs_loc1              RESULT_SET_LOCATOR VARYING ;
    -- result set row:
    DECLARE rownum               INTEGER ;
    DECLARE macro                CHAR(9) ;        -- not used
    DECLARE parameter            VARCHAR(40) ;
    DECLARE install_panel        VARCHAR(8) ;     -- not used
    DECLARE install_field        VARCHAR(40) ;    -- not used
    DECLARE install_location     VARCHAR(12) ;    -- not used
    DECLARE value                VARCHAR(2048) ;
    DECLARE additional_info      VARCHAR(200) ;   -- not used
    -- other variables:
    DECLARE rownum_char          CHAR(10) ;
    DECLARE at_end               SMALLINT ;
    DECLARE start_timestamp      TIMESTAMP ;      -- optional
    DECLARE duration             DECIMAL(20,6) ;  -- optional
    DECLARE duration_char        VARCHAR(100) ;   -- optional
    --
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsa_helper() (3/5)

```
IF ( i_db2_member           IS NULL
     OR LENGTH(i_db2_member) = 0
     OR i_db2_member         = ' ' )
  THEN SET db2_member = NULL ;
ELSE SET db2_member = i_db2_member ;
END IF ;
SET start_timestamp = CURRENT_TIMESTAMP ;
-- initialize (empty) array variable:
SET zParms_array_var
    = ARRAY_DELETE(zParms_array_var) ;
CALL SYSPROC.ADMIN_INFO_SYSPARM
     ( db2_member , retcd , err_msg ) ;
IF SQLCODE = 466 THEN
  ASSOCIATE LOCATORS (rs_loc1 )
            WITH PROCEDURE SYSPROC.ADMIN_INFO_SYSPARM ;
  ALLOCATE c1 CURSOR FOR RESULT SET rs_loc1 ;
  FETCH c1 INTO rownum , macro
      , parameter
      , install_panel , install_field  , install_location
      , value  , additional_info  ;
  SET zParms_array_var [parameter] = value ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsa_helper() (4/5)

```
  SET at_end = 0 ;
  WHILE at_end = 0 DO
    FETCH c1 INTO rownum, macro, parameter
        , install_panel, install_field, install_location
        , value, additional_info    ;
    IF ( SQLCODE = 100 ) THEN SET at_end = 1 ;
    END IF ;
    SET zParms_array_var [parameter] = value ;
    END WHILE ;
  CLOSE  c1 ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsa_helper() (5/5)

```
    -- add additional duration-info:
    SET duration = CURRENT_TIMESTAMP - start_timestamp ;
    SET duration_char = SUBSTR(DIGITS(duration),11,2)
                        CONCAT ' min ' CONCAT
                        SUBSTR(DIGITS(duration),13,2)
                        CONCAT '.'     CONCAT
                        SUBSTR(DIGITS(duration),15,6)
                        CONCAT ' sec' ;
    SET zParms_array_var ['*** Duration or helper UDF: ***']
        = duration_char ;
    -- end of additional duration info -
    RETURN zParms_array_var ;
  ELSE
    SET zParms_array_var ['Error: ']
        = 'NO +466 (Result Set) returned' ;
    RETURN zParms_array_var ;
  END IF ;
END  -- of BEGIN
#
```

© Walter E. Huth, 2021-2024

Create SQL Table UDF zParmsa()

```
CREATE FUNCTION zParmsa ( i_db2_member VARCHAR(8) )
   RETURNS TABLE ( parameter  VARCHAR(  40)
                 , value      VARCHAR(2048)
                 )
   LANGUAGE SQL
   RETURN
     SELECT zp.*
       FROM UNNEST(zParmsa_helper( i_db2_member ))
                                  AS zp (parameter, value)
#
```

1. zParmsa_helper() returns an associative array;
2. UNNEST() generates a relational table (with 2 columns) thereof

© Walter E. Huth, 2021-2024

Select from SQL Table UDF zParmsa()

```
SELECT zp.*
  FROM TABLE (zParmsa('')) zp       -- optional input: Db2 member name
 WHERE parameter LIKE '%COPY%D%'
    OR parameter LIKE '***%'        -- displaying additional information
 ORDER BY parameter                 -- not necessary: array already sorted
#
```

```
---------+---------+---------+---------+---------+---------+---------+
PARAMETER                                VALUE
---------+---------+---------+---------+---------+---------+---------+
*** Duration or helper UDF: ***          00 min 00.248391 sec
FCCOPYDDN                                HLQ.&DB..&SN..N&DSNUM..&UQ.
FLASHCOPY_LOAD                           NO
FLASHCOPY_REBUILD_INDEX                  NO
FLASHCOPY_REORG_INDEX                    NO
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+
```

© Walter E. Huth, 2021-2024

Remarks to array solution

- Array returns **only 2 columns: parameter + value**
- Wanted: additional output columns for zParm: macro, install panel, ...
- Problem: (helper_) function can only return one value
- 1. Idea: use 2-dimensional array

```
-- one common type definition for several SP output columns:
CREATE TYPE ass_array_4zParms AS VARCHAR(2048) ARRAY [INTEGER]
-- one 2-dim. array type (for combining n such arrays):
CREATE TYPE ord_array_4zParms2 AS  ass_array_4zParms  ARRAY [n]
-- In the function body, declare several array variables:
DECLARE zParms_array_var_4Param   ass_array_4zParms ;
DECLARE zParms_array_var_4Value   ass_array_4zParms ;
DECLARE zParms_array_var_4Macro   ass_array_4zParms ;
DECLARE zParms_array_var2         ord_array_4zParms2 ;
-- remark: after 1. UNNEST, 2. UNNEST could handle multiple arrays!
```

rownum

But: only built-in type allowed

- 2. Idea (not pursued for/in this presentation):
  - Up to now, the 2nd column only contains the zParm value
  - Instead, use rownum as array index (=1st column), and
  - Store more data in the 2nd column (param+value+macro..)
  - The Table UDF must extract these parts

© Walter E. Huth, 2021-2024

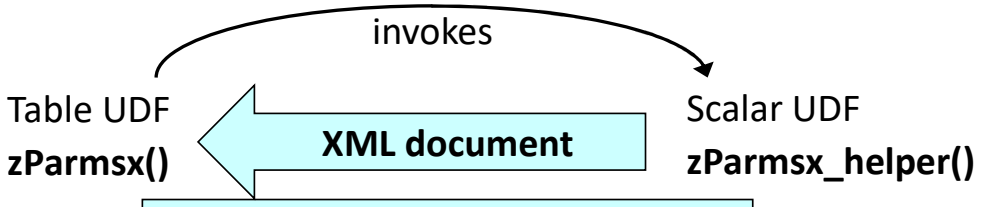
Agenda

- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet
  ← Example: Get one zParm value
- **SQL Table UDF „zParms . ()“**
  ← Example: List all zParm values
  - invokes SQL scalar UDF „zParms ._helper()“
    that calls Stored Procedure [*]
    Data transfer from scalar UDF to table UDF via:
    - Temporary **T**able: „zParms**d/c**()“ ← does not work
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“
  - invokes RESTful service
    that calls Stored Procedure [*]
    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

© Walter E. Huth, 2021-2024

zParmsx() gets data as XML document




```
<ZPARMS>
  <ZP>
    <N>0000000001</N>
    <P>AUDITST</P>
    <V>00000000000000000000000000000000</V>
  </ZP>
  <ZP>
    <N>0000000002</N>
    <P>CONDBAT</P>
    <V>0000001000</V>
  </ZP>
  <ZP>
    <N>0000000003</N>
    <P>CTHREAD</P>
..etc..
</ZPARMS>
```

<N>: rowNum
<P>: Parameter
<V>: Value

special treatment for XML:
"&" replaced by "&":

```
<P>FCCOPYDDN</P>
<V>HLQ.&amp;DB..&amp;SN..N&amp;DSNUM..&amp;UQ.</V>
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsx_helper() (1/5)

```
-- SQL scalar UDF "zParmsx_helper()" that supports SQL table UDF "zParmsx()"
--   input parameter: Db2_member_name (if non data sharing: '' or ' ')
--   calls: sysproc.admin_info_sysparm
--   output: XML document (VARCHAR or CLOB) containing zParm values
--
CREATE FUNCTION zParmsx_helper ( i_db2_member VARCHAR(8) )
  RETURNS VARCHAR(32000)          -- alternative: CLOB(n)

  LANGUAGE SQL
  MODIFIES SQL DATA               -- as udf calls admin_info_sysparm, else -577
  BEGIN
    DECLARE SQLCODE               INTEGER DEFAULT 0 ;
    DECLARE SQLSTATE              CHAR(5) DEFAULT '00000' ;  -- not used
    -- output value:
    DECLARE o_string              VARCHAR(32000) ;           -- or CLOB(n)
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsx_helper() (2/5)

```
    -- parameters for admin_info_sysparm:
    DECLARE db2_member            CHAR(8) ;
    DECLARE retcd                 INTEGER ;
    DECLARE err_msg               VARCHAR(1331) ;
    -- result set locator:
    DECLARE rs_loc1               RESULT_SET_LOCATOR VARYING ;
    -- result set row:
    DECLARE rownum                INTEGER ;
    DECLARE macro                 CHAR(9) ;             -- not used
    DECLARE parameter             VARCHAR(40) ;
    DECLARE install_panel         VARCHAR(8) ;          -- not used
    DECLARE install_field         VARCHAR(40) ;         -- not used
    DECLARE install_location      VARCHAR(12) ;         -- not used
    DECLARE value                 VARCHAR(2048) ;
    DECLARE additional_info       VARCHAR(200) ;        -- not used
    -- other variables:
    DECLARE rownum_char           CHAR(10) ;
    DECLARE at_end                SMALLINT ;
    DECLARE start_timestamp       TIMESTAMP ;           -- optional
    DECLARE duration              DECIMAL(20,6) ;       -- optional
    DECLARE duration_char         VARCHAR(100) ;        -- optional
    --
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsx_helper() (3/5)

```
IF ( i_db2_member           IS NULL
     OR LENGTH(i_db2_member) = 0
     OR i_db2_member         = ' ' )
  THEN SET db2_member = NULL ;
ELSE SET db2_member = i_db2_member ;
END IF ;
SET start_timestamp = CURRENT_TIMESTAMP ;
CALL SYSPROC.ADMIN_INFO_SYSPARM
     ( db2_member , retcd , err_msg ) ;
IF SQLCODE = 466 THEN
  ASSOCIATE LOCATORS (rs_loc1 )
            WITH PROCEDURE SYSPROC.ADMIN_INFO_SYSPARM ;
  ALLOCATE c1 CURSOR FOR RESULT SET rs_loc1 ;
  FETCH c1 INTO rownum , macro
      , parameter
      , install_panel , install_field  , install_location
      , value  , additional_info  ;
  SET rownum_char = DIGITS(rownum) ;
  SET o_string = '<ZPARMS>' ;  -- begin of output value
  SET o_string = o_string CONCAT '<ZP>'
      CONCAT '<N>' CONCAT rownum_char  CONCAT '</N>'
      CONCAT '<P>' CONCAT parameter    CONCAT '</P>'
      CONCAT '<V>' CONCAT value        CONCAT '</V>'
      CONCAT '</ZP>' ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsx_helper() (4/5)

```
SET at_end = 0 ;
WHILE at_end = 0 DO
  FETCH c1 INTO rownum, macro, parameter
      , install_panel, install_field, install_location
      , value, additional_info    ;
  IF ( SQLCODE = 100 ) THEN SET at_end = 1 ;
  END IF ;
  -- special treatment for XML:
  IF ( POSSTR(value,'&') > 0 )   -- FCCOPYDDN contains &
     THEN SET value = REPLACE (value,'&','&amp;') ;
  END IF ;
  SET rownum_char = DIGITS(rownum) ;
  SET o_string = o_string CONCAT '<ZP>'
      CONCAT '<N>' CONCAT rownum_char  CONCAT '</N>'
      CONCAT '<P>' CONCAT parameter    CONCAT '</P>'
      CONCAT '<V>' CONCAT value        CONCAT '</V>'
      CONCAT '</ZP>' ;
  END WHILE ;
CLOSE  c1 ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsx_helper() (5/5)

```
    --  add duration-info:
    SET rownum_char = DIGITS( INTEGER(0) ) ;
    SET duration = CURRENT_TIMESTAMP - start_timestamp ;
    SET duration_char = SUBSTR(DIGITS(duration),11,2)
                        CONCAT ' min ' CONCAT
                        SUBSTR(DIGITS(duration),13,2)
                        CONCAT '.' CONCAT
                        SUBSTR(DIGITS(duration),15,6)
                        CONCAT ' sec' ;
    SET o_string = o_string CONCAT '<ZP>'
        CONCAT '<N>' CONCAT rownum_char              CONCAT '</N>'
        CONCAT '<P>' CONCAT 'Duration of helper UDF' CONCAT '</P>'
        CONCAT '<V>' CONCAT duration_char            CONCAT '</V>'
        CONCAT '</ZP>' ;
    -- end of additional duration info --
    SET o_string = o_string CONCAT '</ZPARMS>' ;
    RETURN o_string ;
  ELSE RETURN 'NO  +466 (RESULT SET)  RETURNED' ;
  END IF ;
END  -- of BEGIN
#
```

© Walter E. Huth, 2021-2024

Create SQL Table UDF zParmsx()

```
CREATE FUNCTION zParmsx ( i_db2_member VARCHAR(8) )
                          -- if non data sharing: '' or ' '
   RETURNS TABLE ( rownum     INTEGER
                 , parameter  VARCHAR(  40)
                 , value      VARCHAR(2048)
                 )
   LANGUAGE SQL
   RETURN
   WITH convert_to_xml_type ( zParms_xml ) AS
   (
     SELECT XMLPARSE(DOCUMENT zParmsx_helper( i_db2_member ) )
            AS zParms_xml
       FROM sysibm.sysdummy1
   )
   SELECT output_tab.*
     FROM convert_to_xml_type
        , XMLTABLE ('$zpx/ZPARMS/ZP'
                    PASSING zParms_xml AS "zpx"
                    COLUMNS rownum    INTEGER       PATH 'N'
                          , parameter VARCHAR(  40) PATH 'P'
                          , value     VARCHAR(2048) PATH 'V' )
          AS output_tab
#
```

1. Scalar UDF **zParmsx_helper()** returns XML document;
2. XMLTABLE() generates relational table thereof

© Walter E. Huth, 2021-2024

Select from SQL Table UDF zParmsx()

```
SELECT rownum                          -- or:  SELECT zp.*
     , parameter
     , value

 FROM TABLE (zParmsx('')) zp           -- optional input: Db2 member name

 WHERE parameter LIKE '%COPY%D%'
    OR rownum = 0                      -- displaying additional information
 ORDER BY parameter
#
```

```
---------+---------+---------+---------+---------+---------+---------+---------+--
     ROWNUM  PARAMETER                                VALUE
---------+---------+---------+---------+---------+---------+---------+---------+--
           0  Duration of helper UDF                  00 min 00.266835 sec
         208  FCCOPYDDN                               HLQ.&DB..&SN..N&DSNUM..&UQ.
         210  FLASHCOPY_LOAD                          NO
         213  FLASHCOPY_REBUILD_INDEX                 NO
         214  FLASHCOPY_REORG_INDEX                   NO
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+--
```

© Walter E. Huth, 2021-2024

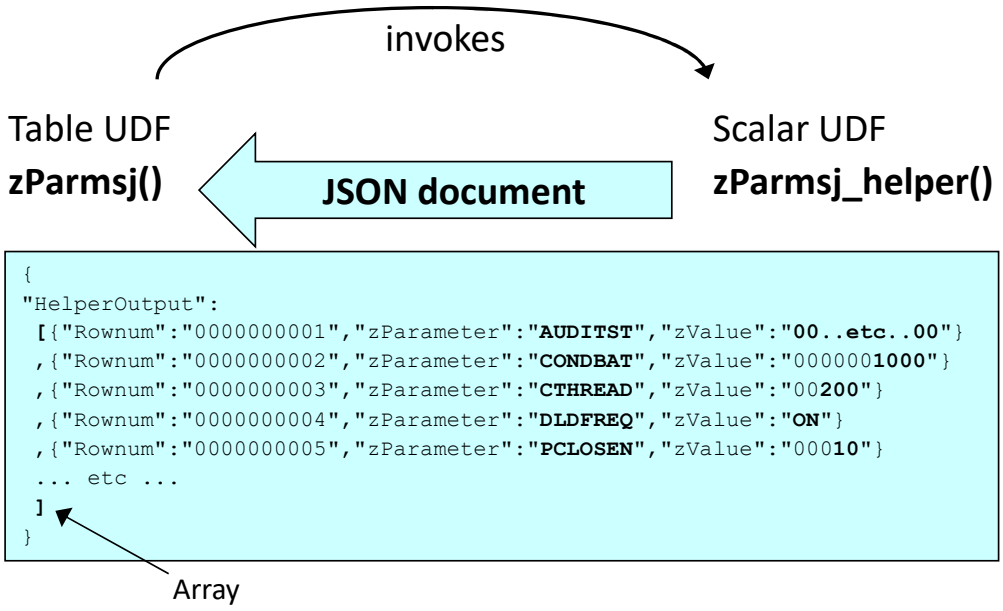
Agenda

- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet
  ← Example: Get one zParm value
- **SQL Table UDF „zParms . ()“**
  ← Example: List all zParm values
  - invokes SQL scalar UDF „zParms ._helper()“
    that calls Stored Procedure [*]
    Data transfer from scalar UDF to table UDF via:
    - Temporary **T**able: „zParms**d/c**()“ ← does not work
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“
  - invokes RESTful service
    that calls Stored Procedure [*]
    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

© Walter E. Huth, 2021-2024

zParmsj() gets data as JSON document



© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsj_helper() (1/4)

```
-- create sql scalar udf zParmsj_helper that supports udf zParmsj
--   input parameter: db2_member_name (if non data sharing: '' or ' ')
--   calls: sysproc.admin_info_sysparm
--   output: JSON document (VARCHAR or CLOB) containing zParm values
--
CREATE FUNCTION zParmsj_helper ( i_db2_member VARCHAR(8) )
  RETURNS VARCHAR(32000)      -- alternative: CLOB(n)
  LANGUAGE SQL
  MODIFIES SQL DATA           -- as udf calls admin_info_sysparm, else -577
  BEGIN
    DECLARE SQLCODE           INTEGER DEFAULT 0 ;
    DECLARE SQLSTATE          CHAR(5) DEFAULT '00000' ;    -- not used
    -- output value:
    DECLARE o_string          VARCHAR(32000) ;             -- or CLOB(n)
    -- and other variables (as before in zParmsx_helper )
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsj_helper() (2/4)

```
IF ( i_db2_member           IS NULL
     OR LENGTH(i_db2_member) = 0
     OR i_db2_member         = ' ' )
  THEN SET db2_member = NULL ;
ELSE SET db2_member = i_db2_member ;
END IF ;
SET start_timestamp = CURRENT_TIMESTAMP ;
CALL SYSPROC.ADMIN_INFO_SYSPARM
     ( db2_member , retcd , err_msg ) ;
IF SQLCODE = 466 THEN
  ASSOCIATE LOCATORS (rs_loc1 )
            WITH PROCEDURE SYSPROC.ADMIN_INFO_SYSPARM ;
  ALLOCATE c1 CURSOR FOR RESULT SET rs_loc1 ;
  FETCH c1 INTO rownum , macro , parameter
      , install_panel , install_field  , install_location
      , value  , additional_info  ;
  SET rownum_char = DIGITS(rownum) ;
  SET o_string = '{"HelperOutput":[' ;            -- begin of output value
  SET o_string = o_string            CONCAT '{"'
                 CONCAT 'Rownum'     CONCAT '":"'
                 CONCAT rownum_char  CONCAT '","'
                 CONCAT 'zParameter' CONCAT '":"'
                 CONCAT parameter    CONCAT '","'
                 CONCAT 'zValue'     CONCAT '":"'
                 CONCAT value        CONCAT '"}'    ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsj_helper() (3/4)

```
  SET at_end = 0 ;
  WHILE at_end = 0 DO
    FETCH c1 INTO rownum, macro, parameter
        , install_panel, install_field, install_location
        , value, additional_info    ;
    IF ( SQLCODE = 100 ) THEN SET at_end = 1 ;
    END IF ;
    SET rownum_char = DIGITS(rownum) ;
    SET o_string = o_string            CONCAT ',{"'
                   CONCAT 'Rownum'     CONCAT '":"'
                   CONCAT rownum_char  CONCAT '","'
                   CONCAT 'zParameter' CONCAT '":"'
                   CONCAT parameter    CONCAT '","'
                   CONCAT 'zValue'     CONCAT '":"'
                   CONCAT value        CONCAT '"}'    ;
    END WHILE ;
  CLOSE  c1 ;
```

© Walter E. Huth, 2021-2024

Create SQL scalar UDF zParmsj_helper() (4/4)

```
    -- add additional duration-info:
    SET rownum_char = DIGITS( INTEGER(0) ) ;
    SET duration = CURRENT_TIMESTAMP - start_timestamp ;
    SET duration_char = SUBSTR(DIGITS(duration),11,2)
                        CONCAT ' min ' CONCAT
                        SUBSTR(DIGITS(duration),13,2)
                        CONCAT '.'     CONCAT
                        SUBSTR(DIGITS(duration),15,6)
                        CONCAT ' sec' ;
    SET o_string = o_string                              CONCAT ',{"'
                   CONCAT 'Rownum'                       CONCAT '":"'
                   CONCAT rownum_char                    CONCAT '","'
                   CONCAT 'zParameter'                   CONCAT '":"'
                   CONCAT 'Duration of helper udf:'      CONCAT '","'
                   CONCAT 'zValue'                       CONCAT '":"'
                   CONCAT duration_char                  CONCAT '"}' ;
    -- end of additional duration info -
    SET o_string = o_string CONCAT ']}' ;           -- end of output value
    RETURN o_string ;
  ELSE RETURN 'NO  +466 (RESULT SET)  RETURNED' ;
  END IF ;
END  -- of BEGIN
#
```

© Walter E. Huth, 2021-2024

Create SQL Table UDF zParmsj()

```
CREATE FUNCTION zParmsj ( i_db2_member VARCHAR(8) )  -- if non data sh.: '' or ' '
   RETURNS table ( rownum     INTEGER
                 , parameter  VARCHAR(  40)
                 , value      VARCHAR(2048)
                 )
   LANGUAGE SQL
   RETURN
   WITH json_table_result ( type, value ) AS
   ( SELECT json_table_result.type, json_table_result.value
       FROM TABLE ( systools.JSON_TABLE
              ( systools.JSON2BSON(zParmsj_helper( i_db2_member ) )
              , 'HelperOutput'
              , 's:2109'    -- z-parameter: 40 + zp-value: 2048 + 3*7 for {"":""}
              )   ) json_table_result
   )
   , json_table_result_in_bson (jt_value_bson) AS
   ( SELECT SYSTOOLS.JSON2BSON(value) AS jt_value_bson
       FROM json_table_result
   )
   SELECT INTEGER(JSON_VAL(jt_value_bson,'Rownum',    's:10'  ))  AS rownum
        ,         JSON_VAL(jt_value_bson,'zParameter','s:40'  )   AS parameter
        ,         JSON_VAL(jt_value_bson,'zValue',    's:2048')   AS value
     FROM json_table_result_in_bson
#
```

1. zParmsj_helper()
returns JSON document;
2. JSON_TABLE()
generates relational table thereof

© Walter E. Huth, 2021-2024

Select from SQL Table UDF zParmsj()

```
SELECT rownum                          -- or:  SELECT zp.*
     , parameter
     , value
  FROM TABLE (zParmsj('')) zp          -- optional input: Db2 member name
 WHERE parameter LIKE '%COPY%D%'
    OR rownum = 0                      -- displaying additional information
 ORDER BY parameter
#
```

```
---------+---------+---------+---------+---------+---------+---------+---------+--
     ROWNUM  PARAMETER                                VALUE
---------+---------+---------+---------+---------+---------+---------+---------+--
          0  Duration of helper udf:                  00 min 00.266835 sec
        208  FCCOPYDDN                                HLQ.&DB..&SN..N&DSNUM..&UQ.
        210  FLASHCOPY_LOAD                           NO
        213  FLASHCOPY_REBUILD_INDEX                  NO
        214  FLASHCOPY_REORG_INDEX                    NO
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+--
```

© Walter E. Huth, 2021-2024

Agenda

- **SQL Scalar UDF „zParm_value()“**
  calls Stored Procedure [*]
  returning a resultSet
  ← Example: Get one zParm value
- **SQL Table UDF „zParms . ()“**
  ← Example: List all zParm values
  - invokes SQL scalar UDF „zParms ._helper()“
    that calls Stored Procedure [*]
    Data transfer from scalar UDF to table UDF via:
    - Temporary **T**able: „zParms**d/c**()“ ← does not work
    - Associative **A**rray: „zParms**a**()“
    - **X**ML document: „zParms**x**()“
    - **J**SON document: „zParms**j**()“
  - invokes RESTful service
    that calls Stored Procedure [*]
    Data transfer from **R**EST service to table UDF via:
    - JSON document: „zParms**r**()“

[*] Example: builtin Stored Procedure SYSPROC.ADMIN_INFO_SYSPARM()

© Walter E. Huth, 2021-2024

Let Db2 generate the JSON document !

For generating the JSON document,

- Forget about scalar UDF zParms**j**_helper() !
- Instead, use a DB2 RESTful service !

  [Service invokes the stored procedure ADMIN_INFO_SYSPARM()]

© Walter E. Huth, 2021-2024

BIND RESTful SERVICE

```
//BINDSRV EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2W)
  --FREE SERVICE("List"."zParms")
    BIND SERVICE("List") NAME ("zParms")       -
        SQLDDNAME(MYSQL) SQLENCODING(EBCDIC)   -
        DESCRIPTION                            -
          ('List all zParms - optionally for certain Db2 member')
  END
//MYSQL    DD *
CALL SYSPROC.ADMIN_INFO_SYSPARM( :DB2_MEMBER, :RETURN_CODE, :MSG   )
//*                              \    IN          OUT        OUT  /
//*                               +--- parameters of service ----+
```

```
GRANT EXECUTE ON PACKAGE "List"."zParms" TO PUBLIC
```

© Walter E. Huth, 2021-2024

Test the new RESTful service

Verify existence:

Browser input: **http://<ddf-ip-addr>:<port>/services/** (*)

```
{"DB2Services":
    [{      }
    ,{      }
    ,{"serviceName":"zParms"
      ,"serviceCollectionID":"List"
      ,"serviceStatus":"started"
      ,"serviceDescription":"List all zParms - opt.. for.. Db2 member"
      ,"serviceProvider":"db2service-1.0"
      ,"serviceURL":"http://<ddf-ip-addr>:<port>/services/List/zParms"}
    ]
}
```

Get description:

Browser input: **http://<ddf-ip-addr>:<port>/services/List/zParms** (*)

```
e.g., description of input parameter:
DB2_MEMBER
  type:
     0:        "null"
     1:        "string"
  maxlenght:   8
  description: "Nullable VARCHAR(8)"
```

(*) prompts for userid & pw

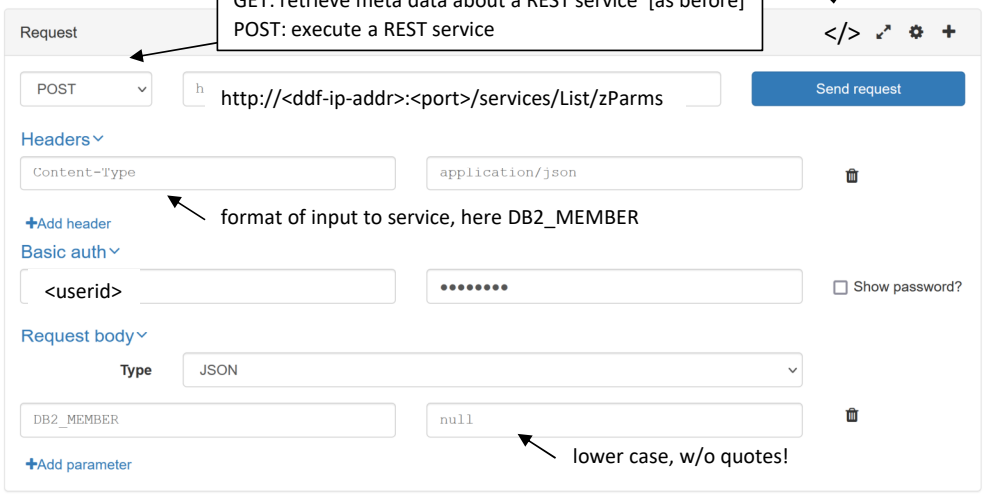
© Walter E. Huth, 2021-2024

Send an http request to Db2

You may use:

- „curl“ (call or see url), running on Windows, Linux, USS,.. [not part of this presentation]
- a browser add-on, e.g., „Rested“ (symbol: </>) for Firefox
- one of Db2‘s SQL functions: http**post**clob(), .. [see later]

GET: retrieve meta data about a REST service [as before]
POST: execute a REST service



© Walter E. Huth, 2021-2024

Browser displays zParms (1/2)

Output of „Rested“:

Response (0.665s) - I http://<ddf-ip-addr>:<port>/services/List/zParms

200 OK

Headers

```
Connection: keep-alive
Content-Length: 65783
Content-Type: application/json; charset=UTF-8
Date: Thu, 25 Nov 2021 10:14:44 GMT
X-Powered-By: DB2 for z/OS
Server: DB2 DDF Native REST, DB2W
Content-Language: en-US
X-Correlation-ID: C0A801FD.ECFC.DAAAA870BE7E
```

The size of the response is greater than 20KB, syntax highlighting has been disabled for performance reasons.

© Walter E. Huth, 2021-2024

Browser displays zParms (1/2)


```
{
  "Output Parameters": {
    "RETURN_CODE": 0,
    "MSG": null
  },
  "ResultSet 1 Output": [
    {
      "ROWNUM": 1,
      "MACRO": "DSN6SYSP",
      "PARAMETER": "AUDITST",
      "INSTALL_PANEL": "DSNTIPN",
      "INSTALL_FIELD": "AUDIT TRACE",
      "INSTALL_LOCATION": "1",
      "VALUE": "00000000000000000000000000000000",
      "ADDITIONAL_INFO": "ONLINE=N"
    },
    {
      "ROWNUM": 2,
      "MACRO": "DSN6SYSP",
      "PARAMETER": "CONDBAT",
      "INSTALL_PANEL": "DSNTIPE",
      "INSTALL_FIELD": "MAX REMOTE CONNECTED",
      "INSTALL_LOCATION": "4",
      "VALUE": "0000001000",
      "ADDITIONAL_INFO": "ONLINE=Y"
    },
```


etc.


```
    {
      "ROWNUM": 353,
      "MACRO": "DB2WIRLM",
      "PARAMETER": "SYSTEM LOCK TABLE LIST ENTRIES",
      "INSTALL_PANEL": "",
      "INSTALL_FIELD": "",
      "INSTALL_LOCATION": "",
      "VALUE": "0000000000",
      "ADDITIONAL_INFO": "ONLINE=N"
    }
  ],
  "StatusCode": 200,
  "StatusDescription": "Execution Successful"
}
```


© Walter E. Huth, 2021-2024

Retrieve JSON result in SPUFI – step 1

```
SELECT DB2XML.HTTPPOSTCLOB(
       CAST('http://<ddf-ip-addr>:<port>/services/List/zParms'
       AS VARCHAR(255))
     , '<httpHeader>
         <header name="Content-Type" value="application/JSON" />
         <header name="Authorization" value="Basic '
         CONCAT
         ( SELECT DB2XML.BASE64ENCODE(CAST
                  ('<userid>:<pw>' AS VARCHAR(64) CCSID 1208))
             FROM SYSIBM.SYSDUMMY1 )
         CONCAT  ' " />
        </httpHeader>'
     , '{"DB2_MEMBER":null}' )
  FROM SYSIBM.SYSDUMMY1
```

service url

input format

authen-tication

input param.

Result (1 row in SPUFI):

```
{"Output Parameters":{"RETURN_CODE":0,"MSG":null},
 "ResultSet 1 Output":
 [{"ROWNUM":1,"MACRO":"DSN6SYSP","PARAMETER":"AUDITST",... }
 ,{"ROWNUM":2,"MACRO":"DSN6SYSP","PARAMETER":"CONDBAT",... }
 ,{"ROWNUM":3,                                          ... }
 ]
}
```

array

© Walter E. Huth, 2021-2024

Retrieve JSON result – step 2

```
{"Output Parameters":{"RETURN_CODE":0,"MSG":null},"ResultSet 1 Output":
[
{"ROWNUM":1,"MACRO":"DSN6SYSP","PARAMETER":"AUDITST",  ... ,
{"ROWNUM":2, ...    }  ]
```

```
WITH service_result_j ( json_col) AS  (   <select-from-previous-page>   )
, service_result_b (bson_col) AS
    ( SELECT SYSTOOLS.JSON2BSON( json_col) as bson_col
        FROM service_result_j )
, value_j (type, value_json) AS
(  SELECT x.*
     FROM service_result_b
        , TABLE ( SYSTOOLS.JSON_TABLE ( bson_col
                                      , 'ResultSet 1 Output'
                                      , 's:4000') ) x )
SELECT * FROM value_j
```

cast

split array into table rows

```
       TYPE  VALUE
---------+---------+---------+---------+---------+---------+---------+----
          3  {ROWNUM:1,MACRO:"DSN6SYSP",PARAMETER:"AUDITST",INSTALL_PANEL:
          3  {ROWNUM:2,MACRO:"DSN6SYSP",PARAMETER:"CONDBAT",INSTALL_PANEL:
          3  {ROWNUM:3,MACRO:"DSN6SYSP",PARAMETER:"CTHREAD",INSTALL_PANEL:
```

© Walter E. Huth, 2021-2024

Retrieve JSON result – step 3

```
       TYPE  VALUE
---------+---------+---------+---------+---------+---------+---------+----
          3  {ROWNUM:1,MACRO:"DSN6SYSP",PARAMETER:"AUDITST",INSTALL_PANEL:
          3  {ROWNUM:2,MACRO:"DSN6SYSP",PARAMETER:"CONDBAT",INSTALL_PANEL:
```

Type 3: JSON sub document

```
....        <--- as above
, value_j (type, value_json) AS (      <select-from-previous-page>    )
, value_b (value_bson) AS
  ( SELECT SYSTOOLS.JSON2BSON(value_json) FROM value_j )
SELECT JSON_VAL(value_bson, 'ROWNUM'    , 'i'     ) AS rownum
     , JSON_VAL(value_bson, 'PARAMETER', 's:40'  ) AS parameter
     , JSON_VAL(value_bson, 'VALUE'     , 's:2048') AS value
  FROM value_b
```

cast

extract element from JSON document

```
     ROWNUM  PARAMETER                        VALUE
---------+---------+---------+---------+---------+-------- .. ..
          1  AUDITST                          0000000000000 .. ..
          2  CONDBAT                          0000001000    .. ..
```

© Walter E. Huth, 2021-2024

Define SQL table UDF zParmsr()

```
CREATE FUNCTION zParmsr ( i_db2_member VARCHAR(8) )
   RETURNS table ( rownum     INTEGER
                 , parameter  VARCHAR(  40)
                 , value      VARCHAR(2048)
                 )
   LANGUAGE SQL
   RETURN
        WITH
          service_result_j (json_col) AS   (   <see-above>   )
        , service_result_b (bson_col) AS   (   <see-above>   )
        , value_j (type, value_json)  AS   (   <see-above>   )
        , value_b (value_bson)        AS   (   <see-above>   )
        SELECT JSON_VAL(...) AS rownum
             , JSON_VAL(...) AS parameter
             , JSON_VAL(...) AS value
          FROM value_b
#
```

© Walter E. Huth, 2021-2024

Select from SQL table UDF "zParmsr()"

```
SELECT rownum                          -- or:  SELECT zp.*
     , parameter
     , value

 FROM TABLE (zParmsr('')) zp           -- optional input: Db2 member name

 WHERE parameter LIKE '%COPY%D%'
 ORDER BY parameter
#
```

```
---------+---------+---------+---------+---------+---------+---------+---------+--
     ROWNUM  PARAMETER                                VALUE
---------+---------+---------+---------+---------+---------+---------+---------+--
        208  FCCOPYDDN                                HLQ.&DB..&SN..N&DSNUM..&UQ.
        210  FLASHCOPY_LOAD                           NO
        213  FLASHCOPY_REBUILD_INDEX                  NO
        214  FLASHCOPY_REORG_INDEX                    NO
DSNE610I NUMBER OF ROWS DISPLAYED IS 5
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+---------+--
```

© Walter E. Huth, 2021-2024

Prerequisites

JSON:

- Server-side UDFs for JSON document access
  (IBM DB2 **Accessories Suite** for z/OS **V4.1 or higher**)
- To create the server-side UDFs, run the DDLs in the DB2 Accessories Suite.
  As usual, a proper WLM environment needs to be configured for these UDFs.

RESTful:

- Software requirements (APARs / PTFs for V12):
  - APAR PI70652 (UI43239, closed 2016) or later
  - PI74515 (closed 2017) UI44784 (caution: JSON parser isssue – abends DB2!)
  - II14827 (Information APAR, e.g., on limitations)
  - PI86868 (UI51748) [for service creation via TSO/Batch]
- **Job DSNTIJRS –** customize and execute –
  for creating the DB2 objects required for REST
  - Database DSNSVCDB, table space DSNSVCTS,
    table SYSIBM.DSNSERVICE, index SYSIBM.DSNSVC01
- Authorize user for DB2 REST services
  - Define a "HTTP protected access profile"
    in an active "RACF DSNR resource class"
  - Authorize the profile via PERMIT
    to access the DB2 REST service
- Recommended: REST client for preferred browser, curl (on Linux, USS,..)

© Walter E. Huth, 2021-2024

Conclusion

- For any stored procedure returning a result set, you can create an **SQL Table UDF** for displaying the result set data in SPUFI, DSNTEP2/4

not examined: max. amount of data

- How? With a
  - helper function, a (preferably SQL) **scalar UDF**, or
  - **REST service**

both call the stored procedure

- The **SQL table function reads from** either:
  - [not successful: from a declared/created temp. table]
  - associative array (but: add'l array type required; only two columns)
  - XML document
  - JSON document
  - Output of REST service

© Walter E. Huth, 2021-2024