Db2 for z/OS:

SQL Table UDF returns result set of Stored Procedure

[via REST or JSON function]

Example: Display zParms in SPUFI

Motivation / Background / Goal / …

- **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 ? – easier!)
- **Problem of SQL Table UDF: Only one fullselect!**

  → no SQL PL, especially: no CALL sp
- **Solution alternative** follows ...
- **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

Agenda

SQL Table UDF „zParms . ()“

Example:
List all
zParm values

- **invokes RESTful service**
  that calls Stored Procedure [*]

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

Alternatives

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

  Data transfer from scalar UDF to table UDF via:
  - **J**SON document: „zParms**j**()“

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

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)

Challenge of SQL Table UDF: only one Fullselect

- **In other words: No SQL PL**
- Bypass the restriction:
  - in a CTE:
    - invoke
      - either: a RESTful service **which contains CALL sp**
      - or: another („helper“) UDF, a scalar UDF **which contains SQL PL, mainly CALL sp**
    - that returns the intended output, a JSON document
  - in the main SELECT:
    - convert the JSON document into a relational result table

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

Fullselect of the SQL Table UDF

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
---------+---------+---------+---------+---------+---------+---------+---------+--
```

Agenda

SQL Table UDF „zParms . ()“

Example:
List all
zParm values

- **invokes RESTful service**
  that calls Stored Procedure [*]

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

Alternatives

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

  Data transfer from scalar UDF to table UDF via:
  - **J**SON document: „zParms**j**()“

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

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
```

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)"
```

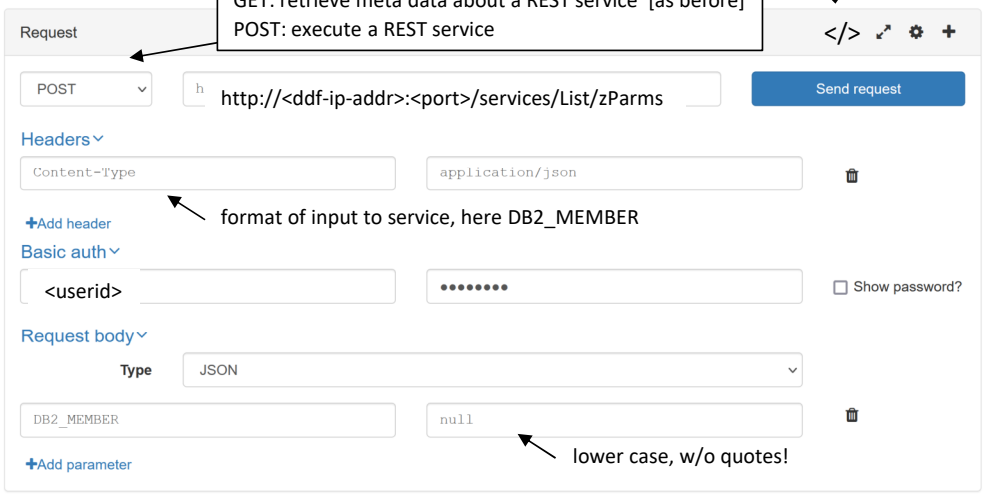
(*) prompts for userid & pw

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



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.

Browser displays zParms (2/2)


```
{
  "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"
}
```

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

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:
```

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    .. ..
```


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
#
```

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
---------+---------+---------+---------+---------+---------+---------+---------+--
```


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,..)

Agenda

SQL Table UDF „zParms . ()“

Example:
List all
zParm values

- **invokes RESTful service**
  that calls Stored Procedure [*]

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

Alternatives

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

  Data transfer from scalar UDF to table UDF via:
  - **J**SON document: „zParms**j**()“

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

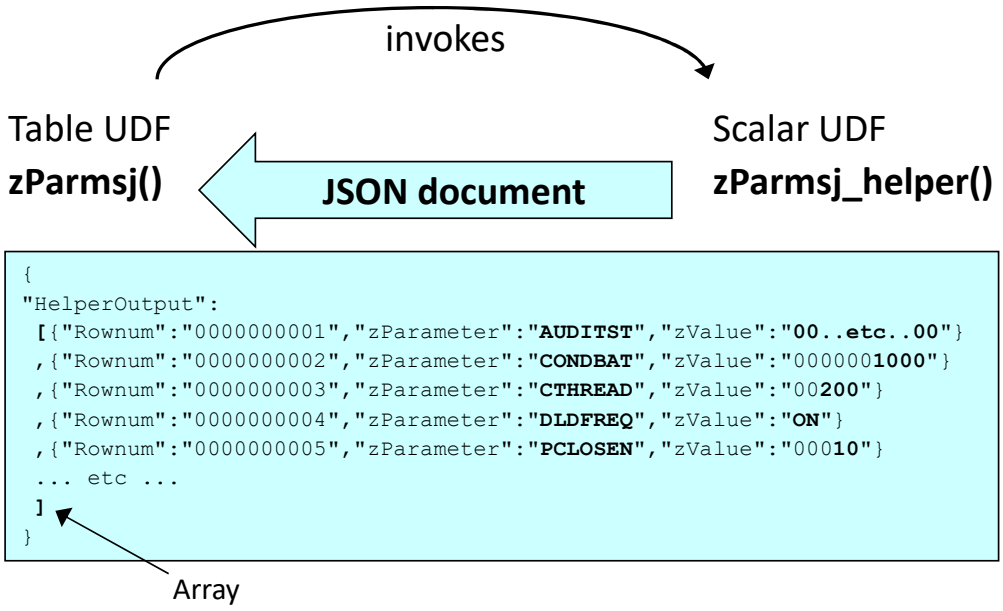
You generate the JSON document !

For generating the JSON document,

- Instead of using a Db2 RESTful service
- Create an SQL scalar UDF „zParmsj_helper()“ !

[Both, the service and the UDF invoke
the stored procedure ADMIN_INFO_SYSPARM()]

zParmsj() gets data as JSON document



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 )
```

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 '"}'    ;
```


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 ;
```

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
#
```


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

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
---------+---------+---------+---------+---------+---------+---------+---------+--
```


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
  - **RESTful service**, or
  - helper function, a (preferably SQL) **scalar UDF** (see below)

both call the stored procedure

- The **SQL table function reads from**:
  - JSON document

Alternate solutions

Instead of data transfer via JSON document,
you can use

- an XML document
  (must handle some special characters)
- an associative array
  (but only with two columns of the result table)
- but not a (declared or created) global temporary table