itoolkit Documentation

Release 1.4.0

Tony Cairns

Jan 11, 2018

Contents:

CHAPTER 1

API

1.1 Toolkit Object

class itoolkit.**iToolKit**(*iparm=0*, *iret=0*, *ids=1*, *irow=1*)

Main iToolKit XMLSERVICE collector and output parser.

Args: iparm (num): include xml node parm output (0-no, 1-yes). iret (num): include xml node return output (0-no, 1-yes). ids (num): include xml node ds output (0-no, 1-yes). irow (num): include xml node row output (0-no, 1-yes).

Returns: iToolKit (obj)

add(*obj*)

Add additional child object.

Args: none

Returns: none

Notes: <?xml version='1.0'?> <xmlservice>

call(*itrans*)

Call xmlservice with accumulated input XML.

Args: itrans (obj): XMLSERVICE transport (iRestCall, iDB2Call, etc.)

Returns: none

clear()

Clear collecting child objects.

Args: none

Returns: (void)

Notes: <?xml version='1.0'?> <xmlservice>

dict_out(*ikey=0*)

return dict output.

Args: ikey (str): select 'key' from {'key':'value'}.

Returns: dict {'key':'value'}

hybrid_out(*ikey=0*)
return hybrid output.

Args: ikey (str): select 'key' from {'key':'value'}.

Returns: hybrid {key:{'data':[list]}}

list_out(*ikey=-1*)
return list output.

Args: ikey (num): select list from index [[0],[1],,,,].

Returns: list [value]

trace_close()
End trace (1.2+)

Args: none

Returns: (void)

trace_hexdump(*itext*)
Write trace hexdump (1.2+) Args:

itext (str): trace text

Returns: (void)

trace_open(*iname='*terminal'*)
Open trace *terminal or file /tmp/python_toolkit_(iname).log (1.2+)

Args: iname (str): trace *terminal or file /tmp/python_toolkit_(iname).log

Returns: (void)

trace_write(*itext*)
Write trace text (1.2+)

Args: itext (str): trace text

Returns: (void)

xml_in()
return raw xml input.

Args: none

Returns: xml

xml_out()
return raw xml output.

Args: none

Returns: xml

1.2 Toolkit Operations

class itoolkit.**iPgm**(*ikey*, *iname*, *iopt={}*)
IBM i XMLSERVICE call *PGM.

Args: ikey (str): XML <ikey>...operation ...</ikey> for parsing output. iname (str): IBM i *PGM or *SRVPGM name iopt (dict): option - dictionay of options (below)

{'error':'on|off|fast'} : optional - XMLSERVICE error choice {'error':'fast'} {'func':'MYFUNC'} : optional - IBM i *SRVPGM function export. {'lib':'mylib'} : optional - IBM i library name {'mode':'opm|ile'} : optional - XMLSERVICE error choice {'mode':'ile'}

Example: iPgm('zzcall','ZZCALL') .addParm(iData('var1','1a','a')) .addParm(iData('var2','1a','b')) .addParm(iData('var3','7p4','32.1234')) .addParm(iData('var4','12p2','33.33')) .addParm(

iDS('var5') .addData(iData('d5var1','1a','a')) .addData(iData('d5var2','1a','b')) .addData(iData('d5var3','7p4','32.1234')) .addData(iData('d5var4','12p2','33.33')))

Returns: iPgm (obj)

Notes:

pgm:

<pgm name="'

[lib="' func="' mode='opm|ile' error='on|off|fast' (1.7.6)]> ... </pgm>

add(*obj*)

Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

addParm(*obj*)

Add a parameter child node.

Args: obj (obj): iData object or iDs object.

Returns: (void)

make()

Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()

Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iSrvPgm**(*ikey*, *iname*, *ifunc*, *iopt={}*)

IBM i XMLSERVICE call *SRVPGM.

Args: ikey (str): XML <ikey>...operation ...</ikey> for parsing output. iname (str): IBM i *PGM or *SRVPGM name ifunc (str): IBM i *SRVPGM function export. iopt (dict): option - dictionay of options (below)

{'error':'on|off|fast'} : optional - XMLSERVICE error choice {'error':'fast'} {'lib':'mylib'} : optional - IBM i library name {'mode':'opm|ile'} : optional - XMLSERVICE error choice {'mode':'ile'}

Example: see iPgm

Returns: iSrvPgm (obj)

Notes:

pgm:

<pgm name="'

[lib="' func="' mode='opm|ile' error='on|off|fast' (1.7.6)]> ... </pgm>

add(*obj*)

Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

addParm(*obj*)

Add a parameter child node.

Args: obj (obj): iData object or iDs object.

Returns: (void)

addRet(*obj*)

Add a return structure child node.

Args: obj (obj): iData object or iDs object.

Returns: (void)

make()

Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()

Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iCmd**(*ikey*, *icmd*, *iopt={}*)

IBM i XMLSERVICE call *CMD not returning *OUTPUT.

Args: ikey (str): XML <ikey>...operation ...</ikey> for parsing output. icmd (str): IBM i command no output (see 5250 command prompt). iopt (dict): option - dictionay of options (below)

{'error':'on|off|fast'} : optional - XMLSERVICE error choice {'error':'fast'} {'exec':cmd|system|rexx'} : optional - XMLSERVICE command execute choice {'exec':'cmd'}

RTVJOBA CCSID(?N) {'exec':'rex'}

Example: iCmd('chglibl', 'CHGLIBL LIBL(XMLSERVICE) CURLIB(XMLSERVICE)') iCmd('rtvjoba', 'RTVJOBA CCSID(?N) OUTQ(?)')

Returns: iCmd (obj)

Notes:

Special commands returning output parameters are allowed. (?) - indicate string return (?N) - indicate numeric return

<cmd [exec='cmd|system|rexx' (default exec='cmd') hex='on' before='cc1/cc2/cc3/cc4' after='cc4/cc3/cc2/cc1' (1.6.8) error='on|off|fast' (1.7.6)]>IBM i command</cmd>

add(*obj*)
Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

make()
Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()
Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iCmd5250**(*ikey*, *icmd*, *iopt={}*)
IBM i XMLSERVICE call 5250 *CMD returning *OUTPUT.

Args: ikey (str): XML <ikey>. . . operation . . . </ikey> for parsing output. icmd (str): IBM i PASE script/utility (see call qp2term). iopt (dict): option - dictionay of options (below)

{'error':'on|off|fast'} : optional - XMLSERVICE error choice {'error':'fast'} {'row':'on|off'} : optional - XMLSERVICE <row>line output</row> choice {'row':'off'}

Example: iCmd5250('dsplibl','dsplibl') iCmd5250('wrkactjob','wrkactjob')

Returns: iCmd5250 (obj)

Notes: This is a subclass of iSh, therefore XMLSERVICE perfoms standard PASE shell popen fork/exec calls.

/QOpenSys/usr/bin/system 'wrkactjob'

Please note, this is a relatively slow operation, use sparingly on high volume web sites.

<sh [rows='on|off' hex='on' before='cc1/cc2/cc3/cc4' after='cc4/cc3/cc2/cc1' (1.7.4) error='on|off|fast' (1.7.6)]>(PASE utility)</sh>

add(*obj*)
Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

make()
Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()
Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iSh**(*ikey*, *icmd*, *iopt={}*)
IBM i XMLSERVICE call PASE utilities.

Args: ikey (str): XML <ikey>...operation ...</ikey> for parsing output. icmd (str): IBM i PASE script/utility (see call qp2term). iopt (dict): option - dictionay of options (below)

{'error':'on|off|fast'} : optional - XMLSERVICE error choice {'error':'fast'} {'row':'on|off'} : optional - XMLSERVICE <row>line output</row> choice {'row':'off'}

Example: iSh('ls /home/xml/master | grep -i xml')

Returns: iSh (obj)

Notes: XMLSERVICE perfoms standard PASE shell popen calls, therefore, additional job will be forked, utilities will be exec'd, and stdout will be collected to be returned.

Please note, this is a relatively slow operation, use sparingly on high volume web sites.

<sh [rows='on|off' hex='on' before='cc1/cc2/cc3/cc4' after='cc4/cc3/cc2/cc1' (1.7.4) error='on|off|fast' (1.7.6)]>(PASE utility)</sh>

add(*obj*)
Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

make()
Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()
Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iXml**(*ixml*)

IBM i XMLSERVICE raw xml input.

Args: ixml (str): custom XML for XMLSERVICE operation.

Example: iXml("<cmd>CHGLIBL LIBL(XMLSERVICE)</cmd>") iXml("<sh>ls /tmp</sh>")

Returns: iXml (obj)

Notes: Not commonly used, but ok when other classes fall short.

add(*obj*)

add input not allowed.

Returns: raise except

make()

Assemble coherent mini dom xml.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()

Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iDS**(*ikey*, *iopt={}*)

Data structure child node for iPgm, iSrvPgm, or nested iDS data structures.

Args: ikey (str): XML <ds ... var="ikey"> for parsing output. iopt (dict): option - dictionay of options (below)

{'dim':'n'} : optional - XMLSERVICE dimension/occurs number. {'dou':'label'} : optional - XMLSERVICE do until label. {'len':'label'} : optional - XMLSERVICE calc length label.

Example: see iPgm

Returns: iDS (obj)

Notes:

pgm data structure:

<ds [dim='n' dou='label' len='label' (1.5.4) data='records' (1.7.5)]>(see <data>)</ds>

add(*obj*)

Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

addData(*obj*)

Add a iData or iDS child node.

Args: obj (obj): iData object or iDs object.

Returns: (void)

make()
Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()
Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

class itoolkit.**iData**(*ikey*, *itype*, *ival*, *iopt={}*)
Data value child node for iPgm, iSrvPgm, or iDS data structures.

Args: ikey (str): XML <data ... var="ikey"> for parsing output. iparm (obj): dom for parameter or return or ds. itype (obj): data type [see XMLSERVICE types, '3i0', ...]. ival (obj): data type value. iopt (dict): option - dictionay of options (below)

{'dim':'n'} : optional - XMLSERVICE dimension/occurs number. {'varying':'on|off|2|4'} : optional - XMLSERVICE varying {'varying':'off'}. {'hex':'on|off'} : optional - XMLSERVICE hex chracter data {'hex':'off'}. {'enddo':'label'} : optional - XMLSERVICE enddo until label. {'setlen':'label'} : optional - XMLSERVICE set calc length label. {'offset':'n'} : optional - XMLSERVICE offset label. {'next':'label'} : optional - XMLSERVICE next offset label (value).

Example: see iPgm

Returns: iData (obj)

Notes:

pgm data elements:

<data type='data types'

[dim='n' varying='on|off|2|4' enddo='label' setlen='label' (1.5.4) offset='label' hex='on|off' before='cc1/cc2/cc3/cc4' after='cc4/cc3/cc2/cc1' (1.6.8) trim='on|off' (1.7.1) next='nextoff' (1.9.2)]>(value)</data>

C types RPG types XMLSERVICE types SQL types ========== ========== ========== ========== int8/byte D myint8 3i 0 <data type='3i0'/> TINYINT (unsupported DB2) int16/short D myint16 5i 0 (4b 0) <data type='5i0'/> SMALLINT int32/int D myint32 10i 0 (9b 0) <data type='10i0'/> INTEGER int64/longlong D myint64 20i 0 <data type='20i0'/> BIGINT uint8/ubyte D myuint8 3u 0 <data type='3u0'/> uint16/ushort D myuint16 5u 0 <data type='5u0'/> uint32/uint D myuint32 10u 0 <data type='10u0'/> uint64/ulonglong D myuint64 20u 0 <data type='20u0'/> char D mychar 32a <data type='32a'/> CHAR(32) varchar2 D myvchar2 32a varying <data type='32a' varying='on'/> VARCHAR(32) varchar4 D myvchar4 32a varying(4) <data type='32a' varying='4'/> packed D mydec 12p 2 <data type='12p2'/> DECIMAL(12,2) zoned D myzone 12s 2 <data type='12s2'/> NUMERIC(12,2) float D myfloat 4f <data type='4f2'/> FLOAT real/double D myreal 8f <data type='8f4'/> REAL binary D mybin (any) <data type='9b'>F1F2F3</data> BINARY hole (no out) D myhole (any) <data type='40h'/> boolean D mybool 1n <data type='4a'/> CHAR(4) time D mytime T timfmt(*iso) <data type='8A'>09.45.29</data> TIME timestamp D mystamp Z <data type='26A'>2011-12-29-12.45.29.000000</data> TIMESTAMP date D mydate D datfmt(*iso) <data type='10A'>2009-05-11</data> DATE

add(*obj*)
Additional mini dom xml child nodes.

Args: obj (iBase) : additional child object

Example: itool = iToolKit() itool.add(

iPgm('zzcall','ZZCALL') <— child of iToolkit .addParm(iData('INCHARA','1a','a')) <— child of iPgm)

Returns: (void)

make()

Assemble coherent mini dom xml, including child nodes.

Args: none

Returns: xml.dom.minidom (obj)

xml_in()

Return XML string of collected mini dom xml child nodes.

Args: none

Returns: XML (str)

1.3 Transports

1.3.1 HTTP Transport

class itoolkit.rest.irestcall.**iRestCall**(*iurl*, *iuid*, *ipwd=0*, *idb2=0*, *ictl=0*, *ipc=0*, *isiz=0*)

Transport XMLSERVICE calls over standard HTTP rest.

Args: iurl (str): XMLSERVICE url (https://common1.frankeni.com:47700/cgi-bin/xmlcgi.pgm). iuid (str): Database user profile name ipwd (str): optional - Database user profile password

– or – env var PASSWORD (export PASSWORD=mypass)

idb2 (str): optional - Database (WRKRDBDIRE *LOCAL) ictl (str): optional - XMLSERVICE control ['*here','*sbmjob'] ipc (str): optional - XMLSERVICE xToolkit job route for *sbmjob ['/tmp/myunique42'] isiz (str): optional - XMLSERVICE expected max XML output size, required for DB2

Example: from itoolkit.rest.irestcall import * itransport = iRestCall(url,user,password)

Returns: none

call(*itool*)

Call xmlservice with accumulated input XML.

Args: itool - iToolkit object

Returns: xml

trace_data()

Return trace driver data.

Args: none

Returns: initialization data

1.3.2 Database Transport

class itoolkit.db2.idb2call.**iDB2Call**(*iuid*, *ipwd=0*, *idb2=0*, *ictl=0*, *ipc=0*, *isiz=0*, *ilib=0*)

Transport XMLSERVICE calls over DB2 connection.

Args: iuid (str): Database user profile name or database connection ipwd (str): optional - Database user profile password

– or – env var PASSWORD (export PASSWORD=mypass)

idb2 (str): optional - Database (WRKRDBDIRE *LOCAL) ictl (str): optional - XMLSERVICE control ['*here','*sbmjob'] ipc (str): optional - XMLSERVICE xToolkit job route for *sbmjob ['/tmp/myunique42'] isiz (int): optional - XMLSERVICE expected max XML output size, required for DB2 ilib (str): optional - XMLSERVICE library compiled (default QXMLSERV)

Example: from itoolkit.db2.idb2call import * itransport = iDB2Call(user,password) – or – conn = ibm_db.connect(database, user, password) itransport = iDB2Call(conn)

Returns: (obj)

call(*itool*)
Call xmlservice with accumulated input XML.

Args: itool - iToolkit object

Returns: xml

trace_data()
Return trace driver data.

Args: none

Returns: initialization data

1.3.3 Direct Memory Transport

class itoolkit.lib.ilibcall.**iLibCall**(*ictl=0*, *ipc=0*, *iccsid=0*, *pccsid=0*)
Transport XMLSERVICE direct job call (within job/process calls).

Args: ictl (str): optional - XMLSERVICE control ['*here','*sbmjob'] ipc (str): optional - XMLSERVICE xToolkit job route for *sbmjob ['/tmp/myunique42'] iccsid (int): optional - XMLSERVICE EBCDIC CCSID [0,37,...] 0 = default jobccsid (1.2+) pccsid (int): optional - XMLSERVICE ASCII CCSID [0,1208,...] 0 = default 1208 (1.2+)

Returns: none

call(*itool*)
Call xmlservice with accumulated input XML.

Args: itool - iToolkit object

Returns: xml

trace_data()
Return trace driver data.

Args: none

Returns: initialization data

CHAPTER 2

Examples

2.1 Calling the DSPSYSSTS CL Command and Displaying Output

```
#                                                                       Bottom
# Type command, press Enter.
# ===> dspsyssts
#                            Display System Status                    LP0364D
#                                                          06/22/15  15:22:28
# % CPU used . . . . . . . . :         .1    Auxiliary storage:
# Elapsed time . . . . . . . :   00:00:01      System ASP . . . . . . :    176.2 G
# Jobs in system . . . . . . :        428      % system ASP used  . . :    75.6481
import config
from itoolkit import *

itool = iToolKit()
itool.add(iCmd5250('dspsyssts', 'dspsyssts'))

# xmlservice
itool.call(config.itransport)

# output
dspsyssts = itool.dict_out('dspsyssts')
if 'error' in dspsyssts:
  print (dspsyssts['error'])
  exit()
else:
  print (dspsyssts['dspsyssts'])
```

2.2 Calling the RTVJOBA CL Command and Getting Output Parameters

```
# RTVJOBA can't issue from command line,
# but works with itoolkit
import config
from itoolkit import *

# modify iToolKit not include row node
itool = iToolKit(iparm=0, iret=0, ids=1, irow=0)
itool.add(iCmd('rtvjoba', 'RTVJOBA USRLIBL(?) SYSLIBL(?) CCSID(?N) OUTQ(?)'))

# xmlservice
itool.call(config.itransport)

# output
rtvjoba = itool.dict_out('rtvjoba')
print (rtvjoba)
if 'error' in rtvjoba:
  print (rtvjoba['error'])
  exit()
else:
  print('USRLIBL = ' + rtvjoba['USRLIBL'])
  print('SYSLIBL = ' + rtvjoba['SYSLIBL'])
  print('CCSID   = ' + rtvjoba['CCSID'])
  print('OUTQ    = ' + rtvjoba['OUTQ'])
```

2.3 Calling the PASE ps Command and Getting Output

```
# > ps -ef
#      UID   PID  PPID   C    STIME    TTY  TIME CMD
#  qsecofr    12    11   0   May 08      -  8:33 /QOpenSys/QIBM/ProdData/JavaVM/jdk60/
↪32bit/jre/lib/ppc/jvmStartPase 566
# qtmhhttp    31     1   0   May 08      -  0:00 /usr/local/zendsvr/bin/watchdog -c /
↪usr/local/zendsvr/etc/watchdog-monitor.ini -s monitor
import config
from itoolkit import *

itool = iToolKit()
itool.add(iSh('ps', 'ps -ef'))

# xmlservice
itool.call(config.itransport)

# output
ps = itool.dict_out('ps')
if 'error' in ps:
  print (ps['error'])
  exit()
else:
  print (ps['ps'])
```

2.4 Tracing to the Terminal

```
import config
from itoolkit import *
itool = iToolKit()
itool.add(
 iPgm('zzcall','ZZCALLNOT')
 .addParm(iData('INCHARA','1a','a'))
 )

# xmlservice write trace log to *terminal
itool.trace_open()
itool.call(config.itransport)
itool.trace_close()

zzcall = itool.dict_out('zzcall')
if 'success' in zzcall:
  print (zzcall['success'])
else:
  print (zzcall['error'])
  exit()
```

2.5 Tracing to a File

```
import config
from itoolkit import *
itool = iToolKit()
itool.add(
 iPgm('zzcall','ZZCALLNOT')
 .addParm(iData('INCHARA','1a','a'))
 )

# xmlservice write trace log to /tmp/python_toolkit_(tonyfile).log
itool.trace_open('tonyfile')
itool.call(config.itransport)
itool.trace_close()

zzcall = itool.dict_out('zzcall')
if 'success' in zzcall:
  print (zzcall['success'])
else:
  print (zzcall['error'])
  exit()
```

2.6 Calling an RPG Program

```
import config
from itoolkit import *
# XMLSERVICE/ZZCALL:
#     D  INCHARA        S              1a
```

```
#     D  INCHARB        S              1a
#     D  INDEC1         S              7p 4
#     D  INDEC2         S             12p 2
#     D  INDS1          DS
#     D   DSCHARA                      1a
#     D   DSCHARB                      1a
#     D   DSDEC1                       7p 4
#     D   DSDEC2                      12p 2
#      *****************************************************************
#      * main(): Control flow
#      *****************************************************************
#     C     *Entry        PLIST
#     C                   PARM                    INCHARA
#     C                   PARM                    INCHARB
#     C                   PARM                    INDEC1
#     C                   PARM                    INDEC2
#     C                   PARM                    INDS1
itool = iToolKit()
itool.add(iCmd('chglibl', 'CHGLIBL LIBL(XMLSERVICE)'))
itool.add(
 iPgm('zzcall','ZZCALL')
 .addParm(iData('INCHARA','1a','a'))
 .addParm(iData('INCHARB','1a','b'))
 .addParm(iData('INDEC1','7p4','32.1234'))
 .addParm(iData('INDEC2','12p2','33.33'))
 .addParm(
  iDS('INDS1')
  .addData(iData('DSCHARA','1a','a'))
  .addData(iData('DSCHARB','1a','b'))
  .addData(iData('DSDEC1','7p4','32.1234'))
  .addData(iData('DSDEC2','12p2','33.33'))
  )
 )

# xmlservice
itool.call(config.itransport)

# output
chglibl = itool.dict_out('chglibl')
if 'success' in chglibl:
  print (chglibl['success'])
else:
  print (chglibl['error'])
  exit()

zzcall = itool.dict_out('zzcall')
if 'success' in zzcall:
  print (zzcall['success'])
  print ("    INCHARA      : " + zzcall['INCHARA'])
  print ("    INCHARB      : " + zzcall['INCHARB'])
  print ("    INDEC1       : " + zzcall['INDEC1'])
  print ("    INDEC2       : " + zzcall['INDEC2'])
  print ("    INDS1.DSCHARA: " + zzcall['INDS1']['DSCHARA'])
  print ("    INDS1.DSCHARB: " + zzcall['INDS1']['DSCHARB'])
  print ("    INDS1.DSDEC1 : " + zzcall['INDS1']['DSDEC1'])
  print ("    INDS1.DSDEC2 : " + zzcall['INDS1']['DSDEC2'])
else:
  print (zzcall['error'])
```

```
  exit()
```

2.7 Calling a Service Program with “Hole” Parameter

```
import config
from itoolkit import *
# Retrieve Hardware Resource List (QGYRHRL, QgyRtvHdwRscList) API
# Service Program: QGYRHR
# Default Public Authority: *USE
# Threadsafe: No
# Required Parameter Group:
#  Output Char(*).............Receiver variable (RHRL0100, RHRL0110)
#  Input Binary(4).............Length of receiver variable
#  Input Char(8)...............Format name
#  Input Binary(4).............Resource category (see hardware resource category)
#  I/O Char(*).................Error code
# RHRL0100 Format
#  BINARY(4)...................Bytes returned
#  BINARY(4)...................Bytes available
#  BINARY(4)...................Number of resources returned
#  BINARY(4)...................Length of resource entry
#  CHAR(*).....................Resource entries
#  These fields repeat for each resource.
#  BINARY(4)...................Resource category
#  BINARY(4)...................Family level
#  BINARY(4)...................Line type
#  CHAR(10)....................Resource name
#  CHAR(4).....................Type number
#  CHAR(3).....................Model number
#  CHAR(1).....................Status
#  CHAR(8).....................System to which adapter is connected
#  CHAR(12)....................Adapter address
#  CHAR(50)....................Description
#  CHAR(24)....................Resource kind (liar, liar, pants on fire ... binary,␣
↪not char)
#  hardware resource category:
#  1  All hardware resources (does not include local area network resources)
#  2  Communication resources
#  3  Local work station resources
#  4  Processor resources
#  5  Storage device resources
#  6  Coupled system adapter resources
#  7  Local area network resources
#  8  Cryptographic resources
#  9  Tape and optical resources
#  10 Tape resources
#  11 Optical resources
itool = iToolKit()
itool.add(
 iSrvPgm('qgyrhr','QGYRHR','QgyRtvHdwRscList')
 .addParm(
  iDS('RHRL0100_t',{'len':'rhrlen'})
  .addData(iData('rhrRet','10i0',''))
  .addData(iData('rhrAvl','10i0',''))
```

```
  .addData(iData('rhrNbr','10i0','',{'enddo':'mycnt'}))
  .addData(iData('rhrLen','10i0',''))
  .addData(iDS('res_t',{'dim':'999','dou':'mycnt'})
           .addData(iData('resCat','10i0',''))
           .addData(iData('resLvl','10i0',''))
           .addData(iData('resLin','10i0',''))
           .addData(iData('resNam','10a',''))
           .addData(iData('resTyp','4a',''))
           .addData(iData('resMod','3a',''))
           .addData(iData('resSts','1a',''))
           .addData(iData('resSys','8a',''))
           .addData(iData('resAdp','12a',''))
           .addData(iData('resDsc','50h','')) # was 50a
           .addData(iData('resKnd','24h','')) # was 24b
           )
 )
 .addParm(iData('rcvlen','10i0','',{'setlen':'rhrlen'}))
 .addParm(iData('fmtnam','10a','RHRL0100'))
 .addParm(iData('rescat','10i0','3')) #  3  Local work station resources
 .addParm(
  iDS('ERRC0100_t',{'len':'errlen'})
  .addData(iData('errRet','10i0',''))
  .addData(iData('errAvl','10i0',''))
  .addData(iData('errExp','7A','',{'setlen':'errlen'}))
  .addData(iData('errRsv','1A',''))
 )
)
# xmlservice
itool.call(config.itransport)
#output
qgyrhr = itool.dict_out('qgyrhr')
if 'success' in qgyrhr:
  print (qgyrhr['success'])
  print ("    Length of receiver variable......" + qgyrhr['rcvlen'])
  print ("    Format name....................." + qgyrhr['fmtnam'])
  print ("    Resource category..............." + qgyrhr['rescat'])
  RHRL0100_t = qgyrhr['RHRL0100_t']
  print ('    RHRL0100_t:')
  print ("      Bytes returned................" + RHRL0100_t['rhrRet'])
  print ("      Bytes available..............." + RHRL0100_t['rhrAvl'])
  print ("      Number of resources returned..." + RHRL0100_t['rhrNbr'])
  print ("      Length of resource entry......." + RHRL0100_t['rhrLen'])
  if int(RHRL0100_t['rhrNbr']) > 0:
    res_t = RHRL0100_t['res_t']
    for rec in res_t:
      print ("        ------------------------------------------------------")
      keys = rec.keys()
      print ("        Resource category............" + rec['resCat'])
      print ("        Family level................" + rec['resLvl'])
      print ("        Line type...................." + rec['resLin'])
      print ("        Resource name..............." + rec['resNam'])
      print ("        Type number................." + rec['resTyp'])
      print ("        Model number................" + rec['resMod'])
      print ("        Status......................" + rec['resSts'])
      print ("        System adapter connected....." + rec['resSys'])
      print ("        Adapter address............." + rec['resAdp'])
      print ("        Description................." + rec['resDsc'])
      print ("        Resource kind..............." + rec['resKnd'])
```

```
else:
  print (qgyrhr['error'])
  exit()
```

2.8 Calling a Service Program

```
import config
from itoolkit import *
# Retrieve Hardware Resource List (QGYRHRL, QgyRtvHdwRscList) API
# Service Program: QGYRHR
# Default Public Authority: *USE
# Threadsafe: No
# Required Parameter Group:
#  Output Char(*).............Receiver variable (RHRL0100, RHRL0110)
#  Input Binary(4).............Length of receiver variable
#  Input Char(8)...............Format name
#  Input Binary(4).............Resource category (see hardware resource category)
#  I/O Char(*).................Error code
# RHRL0100 Format
#  BINARY(4)...................Bytes returned
#  BINARY(4)...................Bytes available
#  BINARY(4)...................Number of resources returned
#  BINARY(4)...................Length of resource entry
#  CHAR(*).....................Resource entries
#  These fields repeat for each resource.
#  BINARY(4)...................Resource category
#  BINARY(4)...................Family level
#  BINARY(4)...................Line type
#  CHAR(10)....................Resource name
#  CHAR(4).....................Type number
#  CHAR(3).....................Model number
#  CHAR(1).....................Status
#  CHAR(8).....................System to which adapter is connected
#  CHAR(12)....................Adapter address
#  CHAR(50)....................Description
#  CHAR(24)....................Resource kind (liar, liar, pants on fire ... binary,␣
↪not char)
#  hardware resource category:
#  1  All hardware resources (does not include local area network resources)
#  2  Communication resources
#  3  Local work station resources
#  4  Processor resources
#  5  Storage device resources
#  6  Coupled system adapter resources
#  7  Local area network resources
#  8  Cryptographic resources
#  9  Tape and optical resources
#  10 Tape resources
#  11 Optical resources
itool = iToolKit()
itool.add(
 iSrvPgm('qgyrhr','QGYRHR','QgyRtvHdwRscList')
 .addParm(
  iDS('RHRL0100_t',{'len':'rhrlen'})
  .addData(iData('rhrRet','10i0',''))
```

```
  .addData(iData('rhrAvl','10i0',''))
  .addData(iData('rhrNbr','10i0','',{'enddo':'mycnt'}))
  .addData(iData('rhrLen','10i0',''))
  .addData(iDS('res_t',{'dim':'999','dou':'mycnt'})
           .addData(iData('resCat','10i0',''))
           .addData(iData('resLvl','10i0',''))
           .addData(iData('resLin','10i0',''))
           .addData(iData('resNam','10a',''))
           .addData(iData('resTyp','4a',''))
           .addData(iData('resMod','3a',''))
           .addData(iData('resSts','1a',''))
           .addData(iData('resSys','8a',''))
           .addData(iData('resAdp','12a',''))
           .addData(iData('resDsc','50a',''))
           .addData(iData('resKnd','24b',''))
           )
 )
 .addParm(iData('rcvlen','10i0','',{'setlen':'rhrlen'}))
 .addParm(iData('fmtnam','10a','RHRL0100'))
 .addParm(iData('rescat','10i0','3')) #  3  Local work station resources
 .addParm(
  iDS('ERRC0100_t',{'len':'errlen'})
  .addData(iData('errRet','10i0',''))
  .addData(iData('errAvl','10i0',''))
  .addData(iData('errExp','7A','',{'setlen':'errlen'}))
  .addData(iData('errRsv','1A',''))
 )
)
# xmlservice
itool.call(config.itransport)
#output
qgyrhr = itool.dict_out('qgyrhr')
if 'success' in qgyrhr:
  print (qgyrhr['success'])
  print ("    Length of receiver variable......" + qgyrhr['rcvlen'])
  print ("    Format name....................." + qgyrhr['fmtnam'])
  print ("    Resource category..............." + qgyrhr['rescat'])
  RHRL0100_t = qgyrhr['RHRL0100_t']
  print ('    RHRL0100_t:')
  print ("      Bytes returned................" + RHRL0100_t['rhrRet'])
  print ("      Bytes available..............." + RHRL0100_t['rhrAvl'])
  print ("      Number of resources returned..." + RHRL0100_t['rhrNbr'])
  print ("      Length of resource entry......." + RHRL0100_t['rhrLen'])
  if int(RHRL0100_t['rhrNbr']) > 0:
    res_t = RHRL0100_t['res_t']
    for rec in res_t:
      print ("        ------------------------------------------------------")
      keys = rec.keys()
      print ("        Resource category............" + rec['resCat'])
      print ("        Family level................" + rec['resLvl'])
      print ("        Line type...................." + rec['resLin'])
      print ("        Resource name..............." + rec['resNam'])
      print ("        Type number................." + rec['resTyp'])
      print ("        Model number................" + rec['resMod'])
      print ("        Status......................" + rec['resSts'])
      print ("        System adapter connected....." + rec['resSys'])
      print ("        Adapter address............." + rec['resAdp'])
      print ("        Description................." + rec['resDsc'])
```

```
      print ("        Resource kind................." + rec['resKnd'])
else:
  print (qgyrhr['error'])
  exit()
```

2.9 Calling a Service Program With an Array Parameter

```
import config
from itoolkit import *
#     D ARRAYMAX        c                   const(999)
#     D dcRec_t         ds                  qualified based(Template)
#     D  dcMyName                    10A
#     D  dcMyJob                   4096A
#     D  dcMyRank                    10i 0
#     D  dcMyPay                     12p 2
#      *****************************************************************
#      * zzarray: check return array aggregate
#      *****************************************************************
#     P zzarray         B                   export
#     D zzarray         PI                  likeds(dcRec_t) dim(ARRAYMAX)
#     D  myName                      10A
#     D  myMax                       10i 0
#     D  myCount                     10i 0
itool = iToolKit()
itool.add(iCmd('chglibl', 'CHGLIBL LIBL(XMLSERVICE)'))
itool.add(
 iSrvPgm('zzarray','ZZSRV','ZZARRAY')
 .addParm(iData('myName','10a','ranger'))
 .addParm(iData('myMax','10i0','8'))
 .addParm(iData('myCount','10i0','',{'enddo':'mycnt'}))
 .addRet(
  iDS('dcRec_t',{'dim':'999','dou':'mycnt'})
  .addData(iData('dcMyName','10a',''))
  .addData(iData('dcMyJob','4096a',''))
  .addData(iData('dcMyRank','10i0',''))
  .addData(iData('dcMyPay','12p2',''))
  )
 )

# xmlservice
itool.call(config.itransport)

# output
# print(itool.xml_out())
chglibl = itool.dict_out('chglibl')
if 'success' in chglibl:
  print (chglibl['success'])
else:
  print (chglibl['error'])
  exit()

zzarray = itool.dict_out('zzarray')
# print(zzarray)
```

```
if 'success' in zzarray:
  print (zzarray['success'])
  print ("    myName       : " + zzarray['myName'])
  print ("    myMax        : " + zzarray['myMax'])
  print ("    myCount      : " + zzarray['myCount'])
  dcRec_t = zzarray['dcRec_t']
  for rec in dcRec_t:
    print ('    dcRec_t:')
    print ("      dcMyName : " + rec['dcMyName'])
    print ("      dcMyJob  : " + rec['dcMyJob'])
    print ("      dcMyRank : " + rec['dcMyRank'])
    print ("      dcMyPay  : " + rec['dcMyPay'])
else:
  print (zzarray['error'])
  exit()
```

2.10 Using *debug to Cause XMLSERVICE to Enter a Message Wait

```
from itoolkit import *
from itoolkit.lib.ilibcall import *

print("********************")
print("********************")
print("Hey user,")
print("Using '*debug' transport parameter allows debug halt before run.")
print ("\n  itransport = iLibCall('*here *debug')\n")
print("Expect qsysopr inquire message, you must answer to continue script.")
print("You may attach a debugger before you answer the inquiry.")
print("\n  dspmsg qsysopr\n")
print("  Reply inquiry message any character.")
print("    From  . . . :   ADC            06/25/15   14:08:07")
print("    Debug client 362262/QSECOFR/QP0ZSPWP")
print("      Reply . . :   c\n")
print("Script continues to run after answer (call PGM, etc.)")
print("********************")
print("********************")

itransport = iLibCall("*here *debug") # i will stop, inquiry message qsysopr
itool = iToolKit()
itool.add(iCmd('chglibl', 'CHGLIBL LIBL(XMLSERVICE)'))
itool.add(
 iPgm('zzcall','ZZCALL')
 .addParm(iData('INCHARA','1a','a'))
 .addParm(iData('INCHARB','1a','b'))
 .addParm(iData('INDEC1','7p4','32.1234'))
 .addParm(iData('INDEC2','12p2','33.33'))
 .addParm(
  iDS('INDS1')
  .addData(iData('DSCHARA','1a','a'))
  .addData(iData('DSCHARB','1a','b'))
  .addData(iData('DSDEC1','7p4','32.1234'))
  .addData(iData('DSDEC2','12p2','33.33'))
  )
 )
```

```
# xmlservice
itool.call(itransport)

# output
chglibl = itool.dict_out('chglibl')
if 'success' in chglibl:
  print (chglibl['success'])
else:
  print (chglibl['error'])
  exit()

zzcall = itool.dict_out('zzcall')
if 'success' in zzcall:
  print (zzcall['success'])
  print ("    INCHARA      : " + zzcall['INCHARA'])
  print ("    INCHARB      : " + zzcall['INCHARB'])
  print ("    INDEC1       : " + zzcall['INDEC1'])
  print ("    INDEC2       : " + zzcall['INDEC2'])
  print ("    INDS1.DSCHARA: " + zzcall['INDS1']['DSCHARA'])
  print ("    INDS1.DSCHARB: " + zzcall['INDS1']['DSCHARB'])
  print ("    INDS1.DSDEC1 : " + zzcall['INDS1']['DSDEC1'])
  print ("    INDS1.DSDEC2 : " + zzcall['INDS1']['DSDEC2'])
else:
  print (zzcall['error'])
  exit()
```

2.11 Using iXml to Get XMLSERVICE Diagnostics

```
import config
from itoolkit import *

# from itoolkit.lib.ilibcall import *
# itransport = iLibCall("*here *debug") # i will stop, inquiry message qsysopr

itool = iToolKit()
itool.add(iCmd('chglibl2', 'CHGLIBL LIBL(QTEMP XMLSERVICE)'))
itool.add(iCmd('chglibl3', 'CHGLIBL LIBL(SOMEBAD42)'))
myxml  = "<diag/>"
itool.add(iXml(myxml))

print(itool.xml_in())


# xmlservice
itool.call(config.itransport)
# itool.call(itransport)

# output
print(itool.xml_out())
diag = itool.dict_out()
if 'version' in diag:
  print ("version   : "+diag['version'])
print ("job       : "+diag['jobnbr']+'/'+diag['jobuser']+'/'+diag['jobname'])
```

```
print ("jobipc    : "+diag['jobipc'])
print ("curuser   : "+diag['curuser'])
print ("ccsid     : "+diag['ccsid'])
print ("dftccsid  : "+diag['dftccsid'])
print ("paseccsid : "+diag['paseccsid'])
print ("syslibl   : "+diag['syslibl'])
print ("usrlibl   : "+diag['usrlibl'])
joblog = diag['joblog'].replace("\n"," ")
cpflist = ""
for word in joblog.split(' '):
  if word[:3] == 'CPF' or word[:3] == 'MCH':
    cpflist += word + " "
    if diag['jobcpf'] == "":
       diag['jobcpf'] = word
print ("jobcpf    : "+diag['jobcpf'] + " ( " + cpflist + ")")
print ("joblog    :\n" + diag['joblog'])
```

2.12 Using iXml to Call a Program with a Varchar Parameter

```
import config
from itoolkit import *
# XMLSERVICE/ZZSRV.ZZVARY:
#     P zzvary          B                   export
#     D zzvary          PI            20A   varying
#     D  myName                       10A   varying
itool = iToolKit()
itool.add(iXml("<cmd var='chglibl'>CHGLIBL LIBL(XMLSERVICE)</cmd>"))
myxml  = "<pgm name='ZZSRV' func='ZZVARY' var='zzvary'>"
myxml += "<parm io='in'>"
myxml += "<data var='myName' type='10A' varying='on'><![CDATA[<Ranger>]]></data>"
myxml += "</parm>"
myxml += "<return>"
myxml += "<data var='myNameis' type='20A' varying='on'><![CDATA[<Mud>]]></data>"
myxml += "</return>"
myxml += "</pgm>"
itool.add(iXml(myxml))

# xmlservice
itool.call(config.itransport)

# output
chglibl = itool.dict_out('chglibl')
if 'success' in chglibl:
  print (chglibl['success'])
else:
  print (chglibl['error'])
  exit()

zzvary = itool.dict_out('zzvary')
if 'success' in zzvary:
  print (zzvary['success'])
  # print ("    myName       : " + zzvary['myName']) ... input only, no output
  print ("    myNameis     : " + zzvary['myNameis'])
else:
  print (zzvary['error'])
```

```
exit()
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index