

---

# **itoolkit Documentation**

***Release 1.7.1-dev***

**Tony Cairns**

**Jun 02, 2022**



---

## Contents

---

|          |                                  |           |
|----------|----------------------------------|-----------|
| <b>1</b> | <b>Usage</b>                     | <b>3</b>  |
| <b>2</b> | <b>Supported Python Versions</b> | <b>5</b>  |
| <b>3</b> | <b>Feature Support</b>           | <b>7</b>  |
| <b>4</b> | <b>Installation</b>              | <b>9</b>  |
| <b>5</b> | <b>Table of Contents</b>         | <b>11</b> |
|          | <b>Python Module Index</b>       | <b>37</b> |
|          | <b>Index</b>                     | <b>39</b> |



itoolkit is a Python interface to the [XMLSERVICE](#) toolkit for the [IBM i](#) platform.



# CHAPTER 1

---

## Usage

---

```
from itoolkit import *
from itoolkit.transport import DatabaseTransport
import ibm_db_dbi

conn = ibm_db_dbi.connect()
itransport = DatabaseTransport(conn)
itool = iToolkit()

itool.add(iCmd5250('wrkactjob', 'WRKACTJOB'))
itool.call(itransport)
wrkactjob = itool.dict_out('wrkactjob')

print(wrkactjob)
```

For more, check out the *Examples*.



---

### Supported Python Versions

---

python-itoolkkit currently supports Python 2.7 and Python 3.4+.

**Warning:** python-itoolkkit 1.x will be the last series to support Python 2.7, 3.4, and 3.5. These versions will no longer be supported in python-itoolkkit 2.0.



## CHAPTER 3

---

### Feature Support

---

- Call ILE programs & service programs
- Call CL Commands
- Call PASE shell commands



## CHAPTER 4

---

### Installation

---

You can install itoolkit simply using *pip*:

```
python -m pip install itoolkit
```



## 5.1 API Reference

### 5.1.1 Toolkit Object

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

Main iToolkit XMLSERVICE collector and output parser.

**Parameters**

- **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.

**Parameters** **none** –

**Returns** none

**Notes**

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

**call** (*itrans*)

Call xmlservice with accumulated input XML.

**Parameters** **itrans** (*obj*) – XMLSERVICE transport object

**Returns** none

**Raises** `TransportClosedException` – If the transport has been closed.

**clear()**

Clear collecting child objects.

**Parameters** `none` –

**Returns** (void)

## Notes

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

**dict\_out** (*ike*=0)

return dict output.

**Parameters** *ike* (*str*) – select ‘key’ from { ‘key’:’value’ }.

**Returns** ‘value’ }

**Return type** dict { ‘key’

**hybrid\_out** (*ike*=0)

return hybrid output.

**Parameters** *ike* (*str*) – select ‘key’ from { ‘key’:’value’ }.

**Returns** { ‘data’: [list] }

**Return type** hybrid {key

**list\_out** (*ike*=-1)

return list output.

**Parameters** *ike* (*num*) – select list from index [[0],[1],...].

**Returns** list [value]

**trace\_close()**

End trace (1.2+)

**Parameters** `none` –

**Returns** (void)

**trace\_hexdump** (*itext*)

Write trace hexdump (1.2+) :param itext: trace text :type itext: str

**Returns** (void)

**trace\_open** (*iname*=’\*terminal’)

Open trace \*terminal or file /tmp/python\_toolkit\_(iname).log (1.2+)

**Parameters** *iname* (*str*) – trace \*terminal or file /tmp/python\_toolkit\_(iname).log

**Returns** (void)

**trace\_write** (*itext*)

Write trace text (1.2+)

**Parameters** *itext* (*str*) – trace text

**Returns** (void)

**xml\_in()**

return raw xml input.

**Parameters** none –

**Returns** xml

**xml\_out()**

return raw xml output.

**Parameters** none –

**Returns** xml

## 5.1.2 Toolkit Operations

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

IBM i XMLSERVICE call \*PGM.

**Parameters**

- **ikey** (*str*) – XML <ikey>...operation...</ikey> for parsing output.
- **iname** (*str*) – IBM i \*PGM or \*SRVPGM name
- **iopt** (*dict*) – option - dictionary of options (below) {‘error’:‘onlofffast’} : XMLSERVICE error choice {‘error’:‘fast’} {‘func’:‘MYFUNC’} : IBM i \*SRVPGM function export. {‘lib’:‘mylib’} : IBM i library name {‘mode’:‘opmlile’} : XMLSERVICE error choice {‘mode’:‘ile’}

### Example

```
iPgm('zzcall','ZZCALL') .addParm(iData('var1','1a','a')) .addParm(iData('var2','1a','b')) .ad-
dParm(iData('var3','7p4','32.1234')) .addParm(iData('var4','12p2','33.33')) .addParm(
    iDS('var5') .addData(iData('d5var1','1a','a')) .addData(iData('d5var2','1a','b')) .ad-
dData(iData('d5var3','7p4','32.1234')) .addData(iData('d5var4','12p2','33.33')) )
```

**Returns** iPgm (obj)

### Notes

**pgm:**

<pgm name=

[lib= func= mode='opmlile' error='onlofffast' (1.7.6) ]> ... </pgm>

**add** (*obj*)

Additional mini dom xml child nodes.

**Parameters** **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*, *iopt*={})

Add a parameter child node.

**Parameters**

- **obj** (*obj*) – iData object or iDs object.
- **iopt** (*dict*) – options to pass to iParm constructor

**Returns** (void)

**make** ()

Assemble coherent mini dom xml, including child nodes.

**Parameters** none –

**Returns** xml.dom.minidom (obj)

**xml\_in** ()

Return XML string of collected mini dom xml child nodes.

**Parameters** none –

**Returns** XML (str)

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

IBM i XMLSERVICE call \*SRVPGM.

**Parameters**

- **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 - dictionary of options (below) {‘error’:‘onlofffast’} : XMLSERVICE error choice {‘error’:‘fast’} {‘lib’:‘mylib’} : IBM i library name {‘mode’:‘opmlile’} : XMLSERVICE error choice {‘mode’:‘ile’}

## Example

see iPgm

**Returns** iSrvPgm (obj)

## Notes

**pgm:**

<pgm name=”

[lib=” func=” mode=’opmlile’ error=’onlofffast’ (1.7.6) ]> ... </pgm>

**add** (*obj*)

Additional mini dom xml child nodes.

**Parameters** **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*, *iopt*={})

Add a parameter child node.

### Parameters

- **obj** (*obj*) – iData object or iDs object.
- **iopt** (*dict*) – options to pass to iParm constructor

**Returns** (void)

**addRet** (*obj*)

Add a return structure child node.

**Parameters** **obj** (*obj*) – iData object or iDs object.

**Returns** (void)

**make** ()

Assemble coherent mini dom xml, including child nodes.

**Parameters** **none** –

**Returns** xml.dom.minidom (*obj*)

**xml\_in** ()

Return XML string of collected mini dom xml child nodes.

**Parameters** **none** –

**Returns** XML (str)

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

IBM i XMLSERVICE call \*CMD not returning \*OUTPUT.

### Parameters

- **ikey** (*str*) – XML <ikey>...operation...</ikey> for parsing output.
- **icmd** (*str*) – IBM i command no output (see 5250 command prompt).
- **iopt** (*dict*) – option - dictionary of options (below) {'error':'onlofflfast'} : XMLSERVICE error option {'exec':cmdsystemlrex'} : XMLSERVICE command type {'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='onloff|fast' (1.7.6) ]>IBM i command</cmd>

**add** (*obj*)

Additional mini dom xml child nodes.

**Parameters** *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.

**Parameters** *none* –

**Returns** xml.dom.minidom (*obj*)

**xml\_in** ()

Return XML string of collected mini dom xml child nodes.

**Parameters** *none* –

**Returns** XML (*str*)

**class** itoolkit.iCmd5250 (*ikey, icmd, iopt={}*)

IBM i XMLSERVICE call 5250 \*CMD returning \*OUTPUT.

**Parameters**

- **ikey** (*str*) – XML <ikey>... operation ...</ikey> for parsing output.
- **icmd** (*str*) – IBM i PASE script/utility (see call qp2term).
- **iopt** (*dict*) – option - dictionary of options (below) { 'error': 'onloff|fast' } : XMLSERVICE error choice { 'error': 'fast' } { 'row': 'onloff' } : XMLSERVICE wrap line in row tag? { 'row': 'off' }

## Example

```
iCmd5250('dsplbl','dsplbl') 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='onloff' hex='on' before='cc1/cc2/cc3/cc4' after='cc4/cc3/cc2/cc1' (1.7.4) error='onlofflfast'
(1.7.6) ]>(PASE utility)</sh>
```

**add** (*obj*)

Additional mini dom xml child nodes.

**Parameters** *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.

**Parameters** *none* –

**Returns** xml.dom.minidom (*obj*)

**xml\_in** ()

Return XML string of collected mini dom xml child nodes.

**Parameters** *none* –

**Returns** XML (str)

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

IBM i XMLSERVICE call PASE utilities.

**Parameters**

- **ikey** (*str*) – XML <ikey>...operation...</ikey> for parsing output.
- **icmd** (*str*) – IBM i PASE script/utility (see call qp2term).
- **iopt** (*dict*) – option - dictionary of options (below) { 'error': 'onlofflfast' } : XMLSERVICE error choice { 'error': 'fast' } { 'row': 'onloff' } : XMLSERVICE wrap line in row tag? { 'row': 'off' }

## Example

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

**Returns** iSh (*obj*)

## Notes

XMLSERVICE performs 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='onloff' hex='on' before='cc1/cc2/cc3/cc4' after='cc4/cc3/cc2/cc1' (1.7.4) error='onlofffast'
(1.7.6) ]>(PASE utility)</sh>
```

**add** (*obj*)

Additional mini dom xml child nodes.

**Parameters** *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.

**Parameters** *none* –

**Returns** xml.dom.minidom (*obj*)

**xml\_in** ()

Return XML string of collected mini dom xml child nodes.

**Parameters** *none* –

**Returns** XML (*str*)

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

IBM i XMLSERVICE raw xml input.

**Parameters** *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.

**Parameters** *none* –

**Returns** xml.dom.minidom (obj)

**xml\_in()**

Return XML string of collected mini dom xml child nodes.

**Parameters** *none* –

**Returns** XML (str)

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

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

**Parameters**

- **ikey** (*str*) – XML <ds ... var="ikey"> for parsing output.
- **iopt** (*dict*) – option - dictionary of options (below) { 'dim': 'n' } : XMLSERVICE dimension/occurs number. { 'dou': 'label' } : XMLSERVICE do until label. { 'len': 'label' } : 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.

**Parameters** **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.

**Parameters** **obj** (*obj*) – iData object or iDs object.

**Returns** (void)

**make** ()

Assemble coherent mini dom xml, including child nodes.

**Parameters** *none* –

**Returns** xml.dom.minidom (obj)

**xml\_in()**

Return XML string of collected mini dom xml child nodes.

**Parameters** none –

**Returns** XML (str)

**class** itoolkit.iData (ikey, itype, ival="", iopt={})

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

**Parameters**

- **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 - dictionary of options (below) { 'dim': 'n' } : XMLSERVICE dimension/occurs number. { 'varying': 'onloff12|4' } : XMLSERVICE varying { 'varying': 'off' }. { 'hex': 'onloff' } : XMLSERVICE hex character data { 'hex': 'off' }. { 'enddo': 'label' } : XMLSERVICE enddo until label. { 'setlen': 'label' } : XMLSERVICE set calc length label. { 'offset': 'n' } : XMLSERVICE offset label. { 'next': 'label' } : XMLSERVICE next offset label (value).

## Example

see iPgm

**Returns** iData (obj)

## Notes

**pgm data elements:**

**<data type='data types'**

[**dim='n'** varying='onloff12|4' enddo='label' setlen='label' (1.5.4) offset='label' hex='onloff' (1.6.8) before='cc1/cc2/cc3/cc4' (1.6.8) after='cc4/cc3/cc2/cc1' (1.6.8) trim='onloff' (1.7.1) next='nextoff' (1.9.2) ]>(value)</data>

For more info on data types you can use, refer to <http://yips.idevcloud.com/wiki/index.php/XMLService/DataTypes>

Changed in version 1.6.3: *ival* is now optional and supports non-string parameters.

**add** (obj)

Additional mini dom xml child nodes.

**Parameters** **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.

**Parameters** none –

**Returns** xml.dom.minidom (obj)

**xml\_in()**

Return XML string of collected mini dom xml child nodes.

**Parameters** none –

**Returns** XML (str)

### 5.1.3 Transports

**class** itoolkit.transport.**XmlServiceTransport** (ctl='\*here \*cdata', ipc='\*na')

XMLSERVICE transport base class

**Parameters**

- **ctl** (str) – XMLSERVICE control options, see <http://yips.idevcloud.com/wiki/index.php/XMLService/XMLSERVICEQuick#ctl>
- **ipc** (str) – An XMLSERVICE ipc key for stateful connections, see <http://yips.idevcloud.com/wiki/index.php/XMLService/XMLSERVICEConnect>

**call** (tk)

Call XMLSERVICE with accumulated actions

**Parameters** tk (iToolkit) – An iToolkit object

**Returns** The XML returned from XMLSERVICE

**Return type** str

**close()**

Close the connection now rather than when `__del__()` is called.

The transport will be unusable from this point forward and a `TransportClosedException` exception will be raised if any operation is attempted with the transport.

### HTTP Transport

**class** itoolkit.transport.**HttpTransport** (url, user, password, database='\*LOCAL', \*\*kwargs)

Call XMLSERVICE using FastCGI endpoint

For more information, refer to <http://yips.idevcloud.com/wiki/index.php/XMLService/XMLSERVICEGeneric>

**Parameters**

- **url** (str) – XMLSERVICE FastCGI endpoint eg. <https://example.com/cgi-bin/xmlcgi.pgm>
- **user** (str) – Database user profile name

- **password**(*str*, *optional*) – Database password
- **database**(*str*, *optional*) – Database name (RDB) to connect to
- **\*\*kwargs** – Base transport options. See *XmlServiceTransport*.

## Example

```
>>> from itoolkit.transport import HttpTransport
>>> endpoint = 'https://example.com/cgi-bin/xmlcgi.pgm'
>>> transport = HttpTransport(endpoint, 'user', 'pass')
```

### call(*tk*)

Call XMLSERVICE with accumulated actions

**Parameters** *tk* (*iToolkit*) – An iToolkit object

**Returns** The XML returned from XMLSERVICE

**Return type** *str*

### close()

Close the connection now rather than when `__del__()` is called.

The transport will be unusable from this point forward and a `TransportClosedException` exception will be raised if any operation is attempted with the transport.

## Database Transport

**class** `itoolkit.transport.DatabaseTransport` (*conn*, *\*\*kwargs*)

Call XMLSERVICE using a database connection

**Parameters**

- **conn** – An active database connection object (PEP-249)
- **schema**(*str*, *optional*) – The XMLSERVICE stored procedure schema to use
- **\*\*kwargs** – Base transport options. See *XmlServiceTransport*.

## Examples

Connecting to XMLSERVICE over ODBC with the default \*LOCAL DSN on IBM i.

```
>>> from itoolkit.transport import DatabaseTransport
>>> import pyodbc
>>> transport = DatabaseTransport(pyodbc.connect('DSN=*LOCAL'))
```

Connecting to XMLSERVICE with `ibm_db_dbi` on IBM i.

```
>>> from itoolkit.transport import DatabaseTransport
>>> import ibm_db_dbi
>>> transport = DatabaseTransport(ibm_db_dbi.connect())
```

### call(*tk*)

Call XMLSERVICE with accumulated actions

**Parameters** *tk* (*iToolkit*) – An iToolkit object

**Returns** The XML returned from XMLSERVICE

**Return type** str

**close()**

Close the connection now rather than when `__del__()` is called.

The transport will be unusable from this point forward and a `TransportClosedException` exception will be raised if any operation is attempted with the transport.

## SSH Transport

**class** `itoolkit.transport.SshTransport` (*sshclient=None*, *\*\*kwargs*)

Transport XMLSERVICE calls over SSH connection.

**Parameters** `sshclient` (*paramiko.SSHClient*) – connected and authenticated connection

### Example

```
>>> from itoolkit.transport import SshTransport
>>> import paramiko
>>> ssh = paramiko.SSHClient()
>>> ssh.set_missing_host_key_policy(paramiko.WarningPolicy())
>>> ssh.connect(host, username="user", password="pass")
>>> transport = SshTransport(ssh)
```

**Warning:** Using `WarningPolicy` is shown only as an example and could lead to security issues. Please refer to the `set_missing_host_key_policy` docs for more info on other policies that may be more appropriate.

**Returns** (obj)

**call** (*tk*)

Call XMLSERVICE with accumulated actions

**Parameters** `tk` (*iToolkit*) – An iToolkit object

**Returns** The XML returned from XMLSERVICE

**Return type** str

**close()**

Close the connection now rather than when `__del__()` is called.

The transport will be unusable from this point forward and a `TransportClosedException` exception will be raised if any operation is attempted with the transport.

## Direct Memory Transport

**class** `itoolkit.transport.DirectTransport` (*\*\*kwargs*)

Call XMLSERVICE directly in-process using `_ILECALL`

**Parameters** *\*\*kwargs* – Base transport options. See *XmlServiceTransport*.

## Example

```
>>> from itoolkit.transport import DirectTransport
>>> transport = DirectTransport()
```

### **call** (*tk*)

Call XMLSERVICE with accumulated actions

**Parameters** *tk* (*iToolkit*) – An iToolkit object

**Returns** The XML returned from XMLSERVICE

**Return type** str

### **close** ()

Close the connection now rather than when `__del__()` is called.

The transport will be unusable from this point forward and a `TransportClosedException` exception will be raised if any operation is attempted with the transport.

---

**Note:** This transport will only work when run on an IBM i system. On other operating systems, calling it will fail with a `RuntimeError`.

---

**Warning:** When using a 64-bit Python, this transport will only work with XMLSERVICE 2.0.1 or higher. When using the system XMLSERVICE in QXMLSERV, the following PTFs are available to fix this problem:

- IBM i 7.4 - SI70669
- IBM i 7.3 - SI70668
- IBM i 7.2 - SI70667

## 5.1.4 Deprecated Transports

**class** `itoolkit.rest.iRESTCall.iRestCall`

Deprecated since version 1.6.0: Use `itoolkit.transport.HttpTransport` instead.

**class** `itoolkit.db2.idb2call.iDB2Call`

Deprecated since version 1.6.0: Use `itoolkit.transport.DatabaseTransport` instead.

**class** `itoolkit.lib.ilibcall.iLibCall`

Deprecated since version 1.6.0: Use `itoolkit.transport.DirectTransport` instead.

## 5.2 Examples

### 5.2.1 Calling the DSPSYSSTS CL Command and Displaying Output

```
#
# Type command, press Enter.
# ==> dspsyssts
#
# Display System Status
#
# Bottom
# LP0364D
```

(continues on next page)

(continued from previous page)

```

#                                     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'])

```

## 5.2.2 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'])

```

## 5.2.3 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

```

(continues on next page)

(continued from previous page)

```
# 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'])
```

## 5.2.4 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()
```

## 5.2.5 Tracing to a File

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

(continues on next page)

(continued from previous page)

```
# 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()
```

## 5.2.6 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)
```

(continues on next page)

(continued from previous page)

```
# 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()
```

## 5.2.7 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
# Threadsafte: 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
```

(continues on next page)

(continued from previous page)

```

# 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':'errrlen'})
    .addData(iData('errRet','10i0',''))
    .addData(iData('errAvl','10i0',''))
    .addData(iData('errExp','7A','',{'setlen':'errrlen'}))
    .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'])

```

(continues on next page)

(continued from previous page)

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

## 5.2.8 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
```

(continues on next page)

(continued from previous page)

```

# 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': 'errrlen'})
    .addData(iData('errRet', '10i0', ''))
    .addData(iData('errAvl', '10i0', ''))
    .addData(iData('errExp', '7A', '', {'setlen': 'errrlen'}))
    .addData(iData('errRsv', '1A', ''))
  )
)
# xmlservice
itool.call(config.itransport)
#output
qgyrhr = itool.dict_out('qgyrhr')
if 'success' in qgyrhr:
    print (qgyrhr['success'])

```

(continues on next page)

(continued from previous page)

```

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

```

### 5.2.9 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         likes(dcRec_t) dim(ARRAYMAX)
#      D  myName             10A
#      D  myMax              10i 0
#      D  myCount            10i 0
itool = iToolKit()
itool.add(iCmd('chglbl', 'CHGLIBL LIBL(XMLSERVICE)'))
itool.add(
    iSrvPgm('zzarray', 'ZZSRV', 'ZZARRAY')
    .addParm(iData('myName', '10a', 'ranger'))
    .addParm(iData('myMax', '10i0', '8'))
)

```

(continues on next page)

(continued from previous page)

```

.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()

```

### 5.2.10 Using \*debug to Cause XMLSERVICE to Enter a Message Wait

```

from itoolkit import *
from itoolkit.transport import DirectTransport

print("*****")
print("*****")
print("Hey user, ")
print("Using '*debug' transport parameter allows debug halt before run.")
print ("\n itransport = DirectTransport('*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.")

```

(continues on next page)

(continued from previous page)

```
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 = DirectTransport("*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()
```

### 5.2.11 Using iXml to Get XMLSERVICE Diagnostics

```
import config
from itoolkit import *

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

itool = iToolKit()
itool.add(iCmd('chglbl12', 'CHGLIBL LIBL(QTEMP XMLSERVICE)'))
itool.add(iCmd('chglbl13', '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 ("jobipcc       : "+diag['jobipcc'])
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['jobcpfl'] == "":
        diag['jobcpfl'] = word
print ("jobcpfl       : "+diag['jobcpfl'] + " ( " + cpflist + ")")
print ("joblog        : \n" + diag['joblog'])
```

### 5.2.12 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='chglbl1'>CHGLIBL LIBL(XMLSERVICE)</cmd>"))
myxml = "<pgm name='ZZSRV' func='ZZVARY' var='zzvary'>"
```

(continues on next page)

(continued from previous page)

```
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()
```

### i

- `itoolkit`, [11](#)
- `itoolkit.db2.idb2call`, [24](#)
- `itoolkit.lib.ilibcall`, [24](#)
- `itoolkit.rest.iRESTcall`, [24](#)
- `itoolkit.transport`, [21](#)



## A

`add()` (*itoolkit.iCmd method*), 16  
`add()` (*itoolkit.iCmd5250 method*), 17  
`add()` (*itoolkit.iData method*), 20  
`add()` (*itoolkit.iDS method*), 19  
`add()` (*itoolkit.iPgm method*), 13  
`add()` (*itoolkit.iSh method*), 18  
`add()` (*itoolkit.iSrvPgm method*), 14  
`add()` (*itoolkit.iToolKit method*), 11  
`add()` (*itoolkit.iXml method*), 18  
`addData()` (*itoolkit.iDS method*), 19  
`addParm()` (*itoolkit.iPgm method*), 14  
`addParm()` (*itoolkit.iSrvPgm method*), 15  
`addRet()` (*itoolkit.iSrvPgm method*), 15

## C

`call()` (*itoolkit.iToolKit method*), 11  
`call()` (*itoolkit.transport.DatabaseTransport method*), 22  
`call()` (*itoolkit.transport.DirectTransport method*), 24  
`call()` (*itoolkit.transport.HttpTransport method*), 22  
`call()` (*itoolkit.transport.SshTransport method*), 23  
`call()` (*itoolkit.transport.XmlServiceTransport method*), 21  
`clear()` (*itoolkit.iToolKit method*), 12  
`close()` (*itoolkit.transport.DatabaseTransport method*), 23  
`close()` (*itoolkit.transport.DirectTransport method*), 24  
`close()` (*itoolkit.transport.HttpTransport method*), 22  
`close()` (*itoolkit.transport.SshTransport method*), 23  
`close()` (*itoolkit.transport.XmlServiceTransport method*), 21

## D

`DatabaseTransport` (*class in itoolkit.transport*), 22  
`dict_out()` (*itoolkit.iToolKit method*), 12  
`DirectTransport` (*class in itoolkit.transport*), 23

## H

`HttpTransport` (*class in itoolkit.transport*), 21  
`hybrid_out()` (*itoolkit.iToolKit method*), 12

## I

`iCmd` (*class in itoolkit*), 15  
`iCmd5250` (*class in itoolkit*), 16  
`iData` (*class in itoolkit*), 20  
`iDB2Call` (*class in itoolkit.db2.idb2call*), 24  
`iDS` (*class in itoolkit*), 19  
`iLibCall` (*class in itoolkit.lib.ilibcall*), 24  
`iPgm` (*class in itoolkit*), 13  
`iRestCall` (*class in itoolkit.rest.irestcall*), 24  
`iSh` (*class in itoolkit*), 17  
`iSrvPgm` (*class in itoolkit*), 14  
`iToolKit` (*class in itoolkit*), 11  
`itoolkit` (*module*), 11  
`itoolkit.db2.idb2call` (*module*), 24  
`itoolkit.lib.ilibcall` (*module*), 24  
`itoolkit.rest.irestcall` (*module*), 24  
`itoolkit.transport` (*module*), 21  
`iXml` (*class in itoolkit*), 18

## L

`list_out()` (*itoolkit.iToolKit method*), 12

## M

`make()` (*itoolkit.iCmd method*), 16  
`make()` (*itoolkit.iCmd5250 method*), 17  
`make()` (*itoolkit.iData method*), 21  
`make()` (*itoolkit.iDS method*), 19  
`make()` (*itoolkit.iPgm method*), 14  
`make()` (*itoolkit.iSh method*), 18  
`make()` (*itoolkit.iSrvPgm method*), 15  
`make()` (*itoolkit.iXml method*), 18

## S

`SshTransport` (*class in itoolkit.transport*), 23

## T

`trace_close()` (*itoolkit.iToolKit method*), 12  
`trace_hexdump()` (*itoolkit.iToolKit method*), 12  
`trace_open()` (*itoolkit.iToolKit method*), 12  
`trace_write()` (*itoolkit.iToolKit method*), 12

## X

`xml_in()` (*itoolkit.iCmd method*), 16  
`xml_in()` (*itoolkit.iCmd5250 method*), 17  
`xml_in()` (*itoolkit.iData method*), 21  
`xml_in()` (*itoolkit.iDS method*), 20  
`xml_in()` (*itoolkit.iPgm method*), 14  
`xml_in()` (*itoolkit.iSh method*), 18  
`xml_in()` (*itoolkit.iSrvPgm method*), 15  
`xml_in()` (*itoolkit.iToolKit method*), 12  
`xml_in()` (*itoolkit.iXml method*), 19  
`xml_out()` (*itoolkit.iToolKit method*), 13  
`XmlServiceTransport` (*class in itoolkit.transport*),  
21