Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by lecacheux.alain on Wed, 18 May 2016 10:58:56 GMT
View Forum Message <> Reply to Message

Le mardi 17 mai 2016 15:54:22 UTC+2, alx a écrit :
> I am trying to use the FDB Python library through IDL in order to access data from some Firebird databases. Everything works very well and transparently as long as data are simple scalar fields in the database.
>
> When the data field to be read is a so called BLOB field (in my case, a binary block of 10240 bytes), the FDB library exposes it as a Python "file-like object", which can be accessed through seek,tell or read functions. Unfortunately, the use of those functions from the imported IDL object looks like to always return empty strings.
>
> More explicitely, after connecting and executing my SQL statement:
>
> IDL> fdb = Python.Import('fdb')
> IDL> con = fdb.connect(dsn=...)
> IDL> cur = con.cursor()
> IDL> cur.execute('SELECT EVT_ID,EVT_TIME,VOLTS FROM SELECT_EVENT(707208)')
>
> I can fetch the data, while specifying that the third field is a blob:
> IDL> print, cur.set_stream_blob('VOLTS')
> IDL> r = cur.fetchone()
>
> The retrieving in IDL is successfull and, as expected, I get 3 field values:
> IDL> help,r
> R LIST <ID=117 NELEMENTS=3>
> IDL> r
> [
> 43630,
> datetime.datetime(2015, 4, 26, 18, 9, 2, 963700),
> <fdb.fbcore.BlobReader object at 0x000000002CCD0FD0>
>]
>
> The third element r[2] is indeed retrieved as a Python callable object, but in trying to read it, I get:
>
> IDL> print, r[2].tell()
> 0
> IDL> q = r[2].read()
> IDL> print, r[2].tell()
> 10240
> IDL> help,q
> Q STRING = ''
>
> The blob was actually entirely read out, but the data did not come through the bridge. Note that the returned 'q' variable should not be a string, since the output of the read() method can be (and

in this case is) an array of binary bytes including null byte.
> Since I am very far from being a Python expert, I might have done a big mistake.
> Does someone have the correct way or any solution ?
> alx.

Looking a bit more deeply at the problem in my previous post, I can summarize it as follows:

Let define a bytes variable in Python:

```
IDL> >>>
>>>  b=bytes([1,2,3,4])
>>>  b
b'\x01\x02\x03\x04'
>>>
IDL> bidl = Python.getattr(Python(), 'b')
IDL> help,bidl
BIDL            STRING    = '    '
IDL> byte(bidl)
  1   2   3   4
```

Looks like ok, but:

```
IDL> >>>
>>>  b=bytes([1,0,3,4])
>>>  b
b'\x01\x00\x03\x04'
>>>
IDL> bidl = Python.getattr(Python(), 'b')
IDL> help, bidl
BIDL            STRING    = ' '
IDL> byte(bidl)
  1
```

In other words, the automatic translation of a Python bytes variable into an IDL string, and not into a byte array, looks like to me a bug.
alx.