Subject: IDL to Python bridge and "file-like" Python object
Posted by lecacheux.alain on Tue, 17 May 2016 13:54:19 GMT
View Forum Message <> Reply to Message

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.

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.

Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by Jim Pendleton on Wed, 18 May 2016 13:15:43 GMT
View Forum Message <> Reply to Message

On Wednesday, May 18, 2016 at 4:58:58 AM UTC-6, alx wrote:
> 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.

Alx,
Looking at the online documentation for BlobReader
http://www.firebirdsql.org/file/documentation/drivers_docume
ntation/python/fdb/reference.html#blobreader, there is this line in the description of the "read" method: "The bytes are returned as a string object. "

One would need to ask why the authors of fdb would choose to return a string instead of a vector numeric type when it is well understood that 0s could be represented as nulls and therefore string terminators. Maybe this isn't how Python constructs strings. I'm not familiar enough with the language to know.

Perhaps there's another Python routine (outside BlobReader) that could be used to convert the Python string (containing nulls) into a byte vector before it is accessed on the IDL side.

Jim P.

Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by lecacheux.alain on Wed, 18 May 2016 16:22:09 GMT
View Forum Message <> Reply to Message

Le mercredi 18 mai 2016 15:15:46 UTC+2, Jim P a écrit :
> On Wednesday, May 18, 2016 at 4:58:58 AM UTC-6, alx wrote:
>> 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.
>
> Alx,
> Looking at the online documentation for BlobReader
http://www.firebirdsql.org/file/documentation/drivers_docume
ntation/python/fdb/reference.html#blobreader, there is this line in the description of the "read" method: "The bytes are returned as a string object. "
>
> One would need to ask why the authors of fdb would choose to return a string instead of a vector numeric type when it is well understood that 0s could be represented as nulls and therefore string terminators. Maybe this isn't how Python constructs strings. I'm not familiar enough with the language to know.
>

> Perhaps there's another Python routine (outside BlobReader) that could be used to convert the Python string (containing nulls) into a byte vector before it is accessed on the IDL side.
>
> Jim P.

Thank you Jim for taking time on this.
The FDB documentation is confusing. I find that the blobreader output is still a bytes variable inside the Python interpreter, but becomes a string variable when "read" in IDL:

```
IDL> rr
<fdb.fbcore.BlobReader object at 0x000000002CD0B080>
IDL> q = rr.read()
IDL> help, q
Q               STRING    = ''
IDL> Python.r2=rr

IDL> >>>
>>> r2
<fdb.fbcore.BlobReader object at 0x000000002CD0B080>
>>> q = r2.read()
>>> type(q)
<class 'bytes'>
```

After the Exelis Python bridge documentation, 'bytes' and 'bytearray' are indeed both converted into IDL strings.

Of course, as you said, it is possible to cast the bytes variable inside Python, then to transfer it to IDL. But it is a bit laborious (and slow ?).

alain.

Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by Fabzi on Thu, 19 May 2016 09:53:16 GMT
View Forum Message <> Reply to Message

On 05/18/2016 06:22 PM, alx wrote:
> Of course, as you said, it is possible to cast the bytes variable inside Python, then to transfer it to IDL. But it is a bit laborious (and slow ?).

Does your package support python3? It seems that the bytarray problems
has something to do with str representations in legacy python.

Cheers,

Fabien

Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by lecacheux.alain on Thu, 19 May 2016 11:39:53 GMT
View Forum Message <> Reply to Message

Le jeudi 19 mai 2016 11:53:19 UTC+2, Fabien a écrit :
> On 05/18/2016 06:22 PM, alx wrote:
>> Of course, as you said, it is possible to cast the bytes variable inside Python, then to transfer it to IDL. But it is a bit laborious (and slow ?).
>
> Does your package support python3? It seems that the bytarray problems
> has something to do with str representations in legacy python.
>
> Cheers,
>
> Fabien

I am using FDB 1.6, Python 3.4 (both quoted as compatible) and IDL 8.5.1.
From my previous post you can check that the FDB blobReader output is a binary bytearray (bytes) within Python, but a string within IDL.
Therefore, the issue looks like to not come from Python but from the IDL/Python bridge, which (incorrectly, I guess) converts Python bytes variable into IDL string (as you can see in the Exelis Python bridge documentation).
This choice is unfortunate because Python bytes variable (afaik, but I am not familiar enough with Python) can contain zero values.
alain.

Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by Jim Pendleton on Thu, 19 May 2016 18:22:24 GMT
View Forum Message <> Reply to Message

On Thursday, May 19, 2016 at 5:39:55 AM UTC-6, alx wrote:
> Le jeudi 19 mai 2016 11:53:19 UTC+2, Fabien a écrit :
>> On 05/18/2016 06:22 PM, alx wrote:
>>> Of course, as you said, it is possible to cast the bytes variable inside Python, then to transfer it to IDL. But it is a bit laborious (and slow ?).
>>
>> Does your package support python3? It seems that the bytarray problems
>> has something to do with str representations in legacy python.
>>
>> Cheers,
>>
>> Fabien
>
> I am using FDB 1.6, Python 3.4 (both quoted as compatible) and IDL 8.5.1.
> From my previous post you can check that the FDB blobReader output is a binary bytearray (bytes) within Python, but a string within IDL.
> Therefore, the issue looks like to not come from Python but from the IDL/Python bridge, which

(incorrectly, I guess) converts Python bytes variable into IDL string (as you can see in the Exelis Python bridge documentation).
> This choice is unfortunate because Python bytes variable (afaik, but I am not familiar enough with Python) can contain zero values.
> alain.

I assume the actual numeric values stored in the blob are not bytes, but are integers, floats, etc.

In this case, you might need to use a utility like Python's struct to reconstitute your data appropriately. Even if you were to get the data into IDL as a BYTARR() you would still need to perform a conversion to INTEGER, FLOAT, etc., AND concern yourself with endianness.

You may be better off by simply letting the existing Python tools do that work for you first.

This reference may be of use, or there may be better solutions:

https://docs.python.org/2/library/struct.html

Jim P.

Subject: Re: IDL to Python bridge and "file-like" Python object
Posted by lecacheux.alain on Thu, 19 May 2016 20:34:05 GMT
View Forum Message <> Reply to Message

Le jeudi 19 mai 2016 20:22:26 UTC+2, Jim P a écrit :
> On Thursday, May 19, 2016 at 5:39:55 AM UTC-6, alx wrote:
>> Le jeudi 19 mai 2016 11:53:19 UTC+2, Fabien a écrit :
>>> On 05/18/2016 06:22 PM, alx wrote:
>>>> Of course, as you said, it is possible to cast the bytes variable inside Python, then to transfer it to IDL. But it is a bit laborious (and slow ?).
>>>
>>> Does your package support python3? It seems that the bytarray problems
>>> has something to do with str representations in legacy python.
>>>
>>> Cheers,
>>>
>>> Fabien
>>
>> I am using FDB 1.6, Python 3.4 (both quoted as compatible) and IDL 8.5.1.
>> From my previous post you can check that the FDB blobReader output is a binary bytearray (bytes) within Python, but a string within IDL.
>> Therefore, the issue looks like to not come from Python but from the IDL/Python bridge, which (incorrectly, I guess) converts Python bytes variable into IDL string (as you can see in the Exelis Python bridge documentation).
>> This choice is unfortunate because Python bytes variable (afaik, but I am not familiar enough with Python) can contain zero values.
>> alain.

>
> I assume the actual numeric values stored in the blob are not bytes, but are integers, floats, etc.
>
> In this case, you might need to use a utility like Python's struct to reconstitute your data appropriately. Even if you were to get the data into IDL as a BYTARR() you would still need to perform a conversion to INTEGER, FLOAT, etc., AND concern yourself with endianness.
>
> You may be better off by simply letting the existing Python tools do that work for you first.
>
> This reference may be of use, or there may be better solutions:
>
> https://docs.python.org/2/library/struct.html
>
> Jim P.

Thank you Jim P. for your helping trick.

My data are indeed made of 2560 floats (forming the 10240 bytes blob), so that:

```
>>>     q = struct.unpack('2560f', r2.read())
```

done on Python side, is doing the job.

However, it would be nice if the translation from bytes/bytearray into string could be improved in further releases of the IDL Python bridge.

alain.
