
Posted by [badhabit](#) on Wed, 24 Jan 2024 16:14:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

```
use test
```

```
drop table if exists tbl
```

```
create table tbl
```

```
(  
    fld_date date  
    , fld_datetime datetime  
    , fld_time time  
    , fld_varchar varchar(50)  
    , fld_int int  
    , fld_float float  
)
```

```
insert into tbl
```

```
select
```

```
'20220914'
```

```
, '20220914 15:12:33'
```

```
, '12:34:22'
```

```
, 'test string'
```

```
, 123
```

```
, 123.45
```

```
#include <iostream>
```

```
#include <string>
```

```
#include <msdasc.h>
```

```
#import "msado15.dll" no_namespace rename("EOF", "adoEOF")
```

```
std::string Type2Str(DataTypeEnum type)
```

```
{  
    std::string sResult;  
    switch (type)  
    {  
        case 0x2000: sResult = "AdArray"; break;  
        case 20: sResult = "adBigInt"; break;  
        case 128: sResult = "adBinary"; break;
```

```

    case 11: sResult = "adBoolean"; break;
    case 8: sResult = "adBSTR"; break;
    case 136: sResult = "adChapter"; break;
    case 129: sResult = "adChar"; break;
    case 6: sResult = "adCurrency"; break;
    case 7: sResult = "adDate"; break;
    case 133: sResult = "adDBDate"; break;
    case 134: sResult = "adDBTime"; break;
    case 135: sResult = "adDBTimeStamp"; break;
    case 14: sResult = "adDecimal"; break;
    case 5: sResult = "adDouble"; break;
    case 0: sResult = "adEmpty"; break;
    case 10: sResult = "adError"; break;
    case 64: sResult = "adFileTime"; break;
    case 72: sResult = "adGUID"; break;
    case 9: sResult = "adIDispatch"; break;
    case 3: sResult = "adInteger"; break;
    case 13: sResult = "adIUnknown"; break;
    case 205: sResult = "adLongVarBinary"; break;
    case 201: sResult = "adLongVarChar"; break;
    case 203: sResult = "adLongVarWChar"; break;
    case 131: sResult = "adNumeric"; break;
    case 138: sResult = "adPropVariant"; break;
    case 4: sResult = "adSingle"; break;
    case 2: sResult = "adSmallInt"; break;
    case 16: sResult = "adTinyInt"; break;
    case 21: sResult = "adUnsignedBigInt"; break;
    case 19: sResult = "adUnsignedInt"; break;
    case 18: sResult = "adUnsignedSmallInt"; break;
    case 17: sResult = "adUnsignedTinyInt"; break;
    case 132: sResult = "adUserDefined"; break;
    case 204: sResult = "adVarBinary"; break;
    case 200: sResult = "adVarChar"; break;
    case 12: sResult = "adVariant"; break;
    case 139: sResult = "adVarNumeric"; break;
    case 202: sResult = "adVarWChar"; break;
    case 130: sResult = "adWChar"; break;
    default: sResult = "err"; break;
}
return sResult;
}

void main()
{
    ColInitialize(NULL);

    _ConnectionPtr pConnection = NULL;
    _RecordsetPtr rs = NULL;

```

```

try
{
    pConnection.CreateInstance(__uuidof(Connection));
    pConnection->CommandTimeout = 0;
    std::string sConnectionString =
"Provider=SQLOLEDB.1;Trusted_Connection=True;Initial Catalog=test;User
ID=sa;Password=sa;Data Source=localhost";
    pConnection->Open(_bstr_t(sConnectionString.c_str()), _bstr_t(""), _bstr_t(""),
adModeUnknown);

    HRESULT hr = rs.CreateInstance(__uuidof(Recordset));
    rs->CursorType = adOpenForwardOnly;
    rs->CursorLocation = adUseServer;
    rs->LockType = adLockReadOnly;
    rs->Open((_bstr_t)"select * from tbl", (IDispatch*)pConnection,
adOpenForwardOnly, adLockReadOnly, adCmdText);

    _variant_t vtFldIdx;
    vtFldIdx.vt = VT_I2;
    for (vtFldIdx.iVal = 0; vtFldIdx.iVal < rs->Fields->GetCount(); vtFldIdx.iVal++)
    {
        std::cout << rs->Fields->GetItem(vtFldIdx)->Type << ": " <<
Type2Str(rs->Fields->GetItem(vtFldIdx)->Type) << std::endl;
    }

    rs->Close();
    pConnection->Close();
}
catch (_com_error& e)
{
    std::cout << "COM err: " << (char*)e.Description() << std::endl;
    if (pConnection && (pConnection->State & adStateOpen) == adStateOpen)
pConnection->Close();
    rs = NULL;
}
catch (...)
{
    std::cout << "err" << std::endl;
    if (pConnection && (pConnection->State & adStateOpen) == adStateOpen)
pConnection->Close();
    rs = NULL;
}
}

```

202: adVarChar
135: adDBTimeStamp

202: adVarChar
200: adVarChar
3: adInteger
5: adDouble

133: adDBDate
135: adDBTimeStamp
145: err
200: adVarChar
3: adInteger
5: adDouble

```
std::string s = (const char*)_bstr_t(val);
```

Posted by [BlackEric](#) on Thu, 25 Jan 2024 07:50:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

Posted by [badhabit](#) on Thu, 25 Jan 2024 08:34:29 GMT
[View Forum Message](#) <> [Reply to Message](#)

Posted by [BlackEric](#) on Thu, 25 Jan 2024 11:42:50 GMT
[View Forum Message](#) <> [Reply to Message](#)

Posted by [SD](#) on Thu, 25 Jan 2024 13:09:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

```
std::string s = (const char*)_bstr_t(val);
```

<https://learn.microsoft.com/en-us/sql/relational-databases/native-client-ole-db-date-time/data-type-support-for-ole-db-date-and-time-improvements?view=sql-server-ver15>

Posted by [badhabit](#) on Fri, 26 Jan 2024 11:21:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

[ative-client-ole-db-date-time/data-type-support-for-ole-db-date-and-time-improvements?view=sql-server-ver15](https://learn.microsoft.com/en-us/sql/relational-databases/native-client-ole-db-date-time/data-type-support-for-ole-db-date-and-time-improvements?view=sql-server-ver15)

```
char*)_bstr_t(vtFldIdx);

if (rs->Fields->GetItem(vtFldIdx)->Type == DBTYPE_DBTIME2)
{
    std::string s = (const char*)_bstr_t(vtFldIdx);
    std::cout << s << std::endl;
}
```

Posted by [SD](#) on Fri, 26 Jan 2024 13:11:33 GMT

[View Forum Message](#) <> [Reply to Message](#)

Posted by [badhabit](#) on Fri, 26 Jan 2024 13:32:33 GMT

[View Forum Message](#) <> [Reply to Message](#)

Posted by [SD](#) on Fri, 26 Jan 2024 22:50:46 GMT

[View Forum Message](#) <> [Reply to Message](#)

<https://learn.microsoft.com/en-us/sql/relational-databases/n>

Posted by [badhabit](#) on Wed, 31 Jan 2024 10:07:32 GMT

[View Forum Message](#) <> [Reply to Message](#)

<https://learn.microsoft.com/en-us/sql/relational-databases/native-client-ole-db-data-types/ssvariant-structure>

```
#include <iostream>
#include <string>

#include <iostream>
#include <string>
#include <c:\Program Files\Microsoft SQL Server\110\SDK\Include\sqlIncli.h>

#import "msado15.dll" no_namespace rename("EOF", "adoEOF")

void main()
{
    ColInitialize(NULL);
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);

    _ConnectionPtr pConnection = NULL;
    _RecordsetPtr rs = NULL;
    try
    {
        pConnection.CreateInstance(__uuidof(Connection));
        pConnection->CommandTimeout = 0;
        std::string sConnectionString = "Provider=SQLNCLI11;Initial Catalog=test;User
ID=sa;Password=sa;Data Source=localhost";
        pConnection->Open(_bstr_t(sConnectionString.c_str()), _bstr_t(""), _bstr_t(""),
adModeUnknown);

        HRESULT hr = rs.CreateInstance(__uuidof(Recordset));
        rs->CursorType = adOpenForwardOnly;
        rs->CursorLocation = adUseServer;
        rs->LockType = adLockReadOnly;
```

```

rs->Open((_bstr_t)"select * from tbl", (IDispatch*)pConnection, adOpenForwardOnly,
adLockReadOnly, adCmdText);

_variant_t vtFldIdx;
vtFldIdx.vt = VT_I2;
for (vtFldIdx.iVal = 0; vtFldIdx.iVal < rs->Fields->GetCount(); vtFldIdx.iVal++)
{
    //std::cout << rs->Fields->GetItem(vtFldIdx)->Type << ": " <<
Type2Str(rs->Fields->GetItem(vtFldIdx)->Type) << std::endl;

    _variant_t val = rs->Collect[vtFldIdx];
    if (val.vt != VT_NULL)
    {
        if (rs->Fields->GetItem(vtFldIdx)->Type == DBTYPE_DBTIME2)
        {
            SSVARIANT ssDst = {0};
            SSVARIANT* pssDst = &ssDst;
            memcpy(&V_SS_TIME2(pssDst).tTime2Val, val, sizeof(DBTIME2)/"cbNative*");
        }
        else
        {
            std::cout << (const char*)_bstr_t(val) << std::endl;
        }
    }
}

rs->Close();
pConnection->Close();
}
catch (_com_error& e)
{
    std::cout << "COM err: " << (char*)e.Description() << std::endl;
    if (pConnection && (pConnection->State & adStateOpen) == adStateOpen)
pConnection->Close();
    rs = NULL;
}
catch (...)
{
    std::cout << "err" << std::endl;
    if (pConnection && (pConnection->State & adStateOpen) == adStateOpen)
pConnection->Close();
    rs = NULL;
}
}

```

Posted by [SD](#) on Wed, 31 Jan 2024 13:02:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

Posted by [badhabit](#) on Wed, 31 Jan 2024 16:24:21 GMT
[View Forum Message](#) <> [Reply to Message](#)

```
_variant_t val = rs->Fields->GetItem(vtFldIdx)->Value;  
if (val.vt != VT_NULL)  
{  
    if (rs->Fields->GetItem(vtFldIdx)->Type == DBTYPE_DBTIME2)  
    {  
        SSVARIANT *pssDst = reinterpret_cast<SSVARIANT*>(val);  
    }  
}
```

Posted by [SD](#) on Wed, 31 Jan 2024 22:36:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

VARIANT.

Posted by [badhabit](#) on Fri, 02 Feb 2024 10:58:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

Posted by [SD](#) on Fri, 02 Feb 2024 13:04:40 GMT

[View Forum Message](#) <> [Reply to Message](#)
