SYNERGY 2015

SEATTLE, WA, USA

SYNERGY 2015
SEATTLE, WA, USA

Understanding and Using UChars

Presented by: John Tuohy

DataAccess WORLDWIDE

The Large Text & Binary Challenge

- What's the best way to read and write large text and binary data to files, databases and properties?
  - Files: Read_Block / Write
  - Database: Get_Field_Value / Set_Field_Value
  - Property: Get / Set

- Strings
  - The file and databases commands expects strings
  - Strings are easy to work with
  - Strings have a size limit
  - Using embedded zeros in strings is quite fragile

- Memory
  - This can be done, but it's not easy
  - You have to do all memory allocation / deallocation / memory copying yourself
  - When things go wrong, they go really wrong

The Large Text & Binary Challenge

- What about using UChar arrays?

UChar data type

- What is a UChar
  - It’s a single byte
  - It can contain a number between 0 and 255
  - Yeah, Byte would have been a better name

```
UChar ucValue
Move 65 to ucValue
```

Char data type

- What is a Char
  - It’s a single signed byte
  - It can contain a number between -128 and +127

```
Char cValue
Move -19 to cValue
```

UChar Arrays

- What is a UChar Array
  - It’s an array of bytes
  - Each array element contains a value of 0 to 255
  - The length of the array is SizeOfArray()

```
UChar[] MyData
Move 65 to MyData[0]
Move 66 to MyData[1]
Move 67 to MyData[2]
Move (SizeOfArray(MyData)) to iLength
```

UChar Arrays and big data

- UChar arrays are good types for dealing with large text data and binary data
  - Can be any size
  - Their length is always known
  - Can contain embedded zeros
  - Can be stored in properties
  - Are easily created, sized, resized and disposed
  - They are bytes – no encoding is applied

UChar Arrays and big data

- Unfortunately, until 18.1, UChar arrays were an island to themselves

- That has changed...

Array Functions and UChar

- Any of the array functions can be used with UChar arrays

- This includes the new AppendArray() function

    - Move (AppendArray(UCharData1, UCharData2)) to UCharData3

UChar Arrays and Strings

- New functions have been created to make it easier to convert data between UChar arrays and strings
    - StringToUCharArray()

```
Move (StringToUCharArray(sStr)) to UCharData
Move (StringToUCharArray("ABC")) to UCharData
```

    - UCharArrayToString()

```
Move (UCharArrayToString(UCharData)) to sStr
Move (UCharArrayToString(UCharData,iLength)) to sStr
Move (UCharArrayToString(UCharData,iLength,iPosition)) to sStr
```

UChar Arrays in 18.1

- UChar arrays can now be used to:
  - Read and write to sequential files
    - Read_Block
    - Write
  - Read and write to database tables
    - Get_Field_Value
    - Set_Field_Value

  - Just use a UChar array data type with these commands and the runtime does the rest

UChar Arrays and Memory

- UChar Arrays can work with memory buffers
- UChar Arrays are stored as a continuous bytes
- The first address is obtained using AddressOf()
- Because you are working with memory it’s more work:
  - You may to allocate / deallocate memory
  - You may need to resize the array
  - You need to do this right
    - You need to make sure you don’t overwrite memory
- This is unchanged in 18.1

UChar Arrays and Memory

```
Function CopyUCharToMemory UChar[] UCData Returns Address
    Address aData
    Integer iLen
    Boolean bOk
    Move (SizeOfArray(UCData)) to iLen
    Move (Alloc(iLen)) to aData
    Move (MemCopy(aData,AddressOf(UCData),iLen)) to bOk
    Function_Return aData
End_Function


Function CopyMemoryToUChar Address aData Integer iLen Returns UChar[]
    UChar[] UCData
    Boolean bOk
    Move (ResizeArray(UCData,iLen)) to UCData
    Move (MemCopy(AddressOf(UCData),aData, iLen)) to bOk
    Function_Return UCData
End_Function
```

Examples