aardio 范例: COM 接口 - 类型转换规则

```
//COM 接口 - 类型转换规则

/****
aardio 支持 COM 原生接口，在标准库的 com.interface 名字空间下可找到很多这类原生接口。
COM 原生接口使用的参数的数据类型与调用原生原生 API 相类似，参考 raw 库相关文档即可。

我们一般所说的 COM 对象并非上面所述的原生接口对象，而是实现了 COM 动态接口的 IDispatch 对象，
com.CreateObject 用于 创建 IDispatch 对象， com.IsObject() 函数用于判断参数是否 IDispatch 对象。
我们一般提到的 “COM 对象”也专指这种动态接口的 IDispatch 对象。

▸ IDispatch 接口类型转换：
----------

IDispatch 接口传参主要使用变体类型（Variant），也就是 aardio 中的 com.Variant 对象。
变体类型（Variant）本质上是一个结构体，并且用一个 vt 字段声明了自己的类型，例如 VT_I4 就表示4字节的整型数值。
例如一个 VT_I4 类型的数值 123，用 aardio 结构体描述就是这样：
{
    WORD vt = 3/*_VT_I4*/;
    WORD r1;
    WORD r2;
    WORD r3;
    ptr lVal = 123;
    ptr r4;
};

在 aardio 中调用 COM 函数，如果 COM 函数明确声明了所需的参数类型，
这时 aardio 会自动将参数转换为合适的 COM 类型。如果 COM 函数没有明确声明参数类型，
默认情况下按如下规则转换：

普通的非数组值会自动转换为 Variant 变体，并将参数值转换为合适的类型，
例如字符串转为 _VT_BSTR 类型,时间转换为 _VT_DATE 类型 …… 这些相互兼容的类型这里不多讲。

主要导致参数类型报错的通常是数值类型参数，我们要特别注意 aardio 对数值的转换规则：
单个整数值默认在 COM 函数中会转为 4 字节 VT_I4( int ) 类型变体，而小数值默认转为 8 字节 _VT_R8（double) 类型变体。

▸ IDispatch 接口数组类型转换：
----------

COM 中的数组则称为安全数组（ SAFEARRAY ），
SAFEARRAY 可以是各种类型的数据组成，例如数值数组，字符串数组，或者由 Variant 对象组成的 Variant 数组。
因为 Variant 对象本身可以承载不同的数据类型，所以 Variant 数组当然也就可以包含各种类型的数组成员了。

纯数值 aardio 数组默认会转为 _VT_R8（double) 类型 SAFEARRAY，要特别注意这里没有使用整型。
纯字符串 aardio 数组默认会转为 _VT_BSTR 类型 SAFEARRAY。
普通 COM 对象（IDispatch 对象） aardio 数组会转为 _VT_DISPATCH 类型 SAFEARRAY。
其他类型 aardio 数组会转为 _VT_VARIANT 类型 SAFEARRAY。

▸ 改变 IDispatch 接口类型转换时的默认类型：
----------

这一般是没有问题的，一般 COM 对象也会自动做兼容的类型转换，
但仍然会出现一些例外，例如我们用 com.CreateObject("ScriptControl") 创建 VBScript 的脚本解释器，
在 VBScript 里就只能识别 Variant 类型的对象，数组也只能为 Variant 类型数组。

当然，我们可以直接调用 com.Variant({1,2,3}) 这种方式创建 Variant 类型的数组，
但这样显然很麻烦，对于这样的 COM 对象，我们要使用 com.SetPreferredArrayType(comObject,0xC/*_VT_VARIANT*/)
将其设置为始终使用 Variant 类型转换参数，可查看封装了 ScriptControl 的标准库 web.script 就是这样做的。

com.SetPreferredArrayType(comObject,0xFFFF/*_VT_ILLEGAL*/) 则会告知 aardio 使用默认规则自动分析数组类型，
这句代码是永远不需要调用的，因为这是 COM 对象的默认设置。注意 com.SetPreferredArrayType() 的作用具有传递性，
COM 对象会将这个函数设置的值传递给他返回的其他 COM 对象。

▸ 明确声明 COM 参数类型：
----------

有一些 COM 对象则恰恰相反，要非常明确的指定数组的类型，
例如函数要求 16位无符号整型的数组，你传 32 位整型数组就会报错，
标准库的 dotNet 基于 COM 接口调用 .Net ，.Net 就有这个特性，需要更严格地匹配参数类型。

aardio 已在 dotNet 支持库实现了比较完美的类型自动转换和自动兼容。
但我们仍然要了解如何在 COM 接口中明确地声明数值或数组的类型。

因为我们可能在其他 COM 对象中遇到这种问题（虽然并不多见），例如 AutoCAD 的接口。

aardio 提供了以下函数可以明确的声明 COM 参数的类型：
-----------------------------------------------------------------
com.byte() 将参数指定的数值或数组声明为 8 位整型数值。
```

```
com.ubyte()  将参数指定的数值或数组声明为 8 位无符号整型数值。
com.word() 将参数指定的数值或数组声明为 16 位整型数值。
com.uword() 将参数指定的数值或数组声明为 16 位无符号整型数值。
com.int() 将参数指定的数值或数组声明为 32 位整型数值。
com.uint() 将参数指定的数值或数组声明为 32 位无符号整型数值。
com.long() 将参数指定的数值或数组声明为 64 位整型数值。
com.ulong() 将参数指定的数值或数组声明为 64 位无符号整型数值。
com.float() 将参数指定的数值或数组声明为 32 位浮点数值。
com.double() 将参数指定的数值或数组声明为 64 位浮点数值。
com.Variant() 将参数指定的值或数组声明为变体类型。
-------------------------------------------------------------------
aardio 中原生类型使用大写表示无符号数值类型，
在以上函数名字中我们在小写的类型名前加上"u" 取代大写以表示无符号类型，

要注意不同编程语言之间的差别：
VB6/VBA 中 Integer 是16位数值，Long 是32位数值，
而在 C# 中 int 是32位数值，long 是64位数值，
更重要的不是类型名字，而是存储长度。
****/

import com;

//创建用于 COM 函数的 16位数值
var v = com.word(32)

//创建用于 COM 函数的 16位数组
var v = com.word({1,2,3})

//创建用于 COM 函数的 Variant 类型变体数组（注意有的参数不允许字符串数组，要求传变体数组）
var v = com.Variant({"test","test2"});

//下面这样写与 com.word({1,2,3}) 的作用是一样的。
var v = com.Variant({1,2,3},2/*_VT_I2*/);

//创建用于 COM 函数的 Variant 类型变体数组，
var v = com.SafeArrayV({1,2,3}); //这个函数返回的不是 Variant 也不是 SAFEARRAY,只是包装对象

//下面这样写与 com.word({1,2,3}) 的作用是一样的，但返回的不是 Variant 对象而是包装对象
var v = com.SafeArrayV({1,2,3},2/*_VT_I2*/);

//下面这样是创建 SAFEARRAY 数组，可以作为普通 aardio 数组使用，也可以作为 COM 参数用
var v  = com.SafeArray(); //COM 对象返回的所有 SAFEARRAY 数组也是这种格式
table.push(v,1,2,3); //实际上 SAFEARRAY 也是普通数组

//也可以在参数中指明类型，下面代码作用类似 com.word({1,2,3})，但返回的是  SAFEARRAY 而非 Variant 对象。
var v  = com.SafeArray(_VT_I2,1,2,3);

/*
COM 函数也可以兼容 raw 库以下函数创建的类型化数值：
-------------------------------------------------------------------
raw.byte()
raw.ubyte()
raw.word()
raw.uword()
raw.int()
raw.uint()
raw.long()
raw.ulong()
raw.float()
raw.double()
-------------------------------------------------------------------
*/
```

Markdown 格式