Vapula Envelope设计

Envelope是Method的参数信封

当Library被加载后，每个Method会创建一个Envelope原型

当需要调用Method时，

Invoker的构造和初始化过程中，会复制一个Envelope

并将该实用副本托管于Invoker的Stack中

Envelope用于Method的用户参数管理

Vapula通过堆（和内存池）管理参数数据

每个Variable包含参数类型、方向（输入、输出、输入或输出）、数据长度

Vapula的Envelope支持所有数据类型。

但是受制于不同语言、运行时对数据类型的定义的差别

Vapula做出以下限制：

除Object和String外，其他类型为基值类型

String是Object的一种特殊细分，作为非基值类型没有本质区别，只是可以使用更简单的API进行操作。

基值类型变量的数据作为一个数组存储

非基值类型变量的数据只能存储为一个独立个体，不能存储为数组

Vapula会识别写入和读取的类型是否是基值类型

根据判定结果，Vapula确定是否可以调用访问方法

同时，Vapula不关心数据的具体存储类型

对于基值类型，Vapula提供一些语法糖（模板函数），

可进行快速单值读取和写入，目的仅仅是简化操作

再次重申，Vapula不检测访问基值类型时使用的数据类型是否匹配

写入字符串时会自动复制并存储副本

投递操作：

投递是Vapula定义的一种行为，表示参数从一个信封的一个变量传递到另一个信封的另一个变量。

默认投递是Deliver，用于精确数据传递，要求类型完全一致。

由于Vapula不关心非基值类型的实际类型信息，所以object类型之间虽然执行了精确数据复制，但是不能保证这个操作是有意义的。

高级的投递方式是CastDeliver，用于带有数据类型转换的数据传递，不要求类型完全一致，但是需要符合转型规则，另外这个API的效率较低。

Object到String：取Object内存地址并通过uint32转换到String复制。

Object到Object，同型，精确复制。

Object到int8、int16、uint8、uint16、real32、real64：转换没有明确意义，直接赋0。

Object到int32、uint32 、int64、uint64：取Object内存地址并通过uint32转换，可能发生截断。

Object到bool：不为null时true

String到String：同型，复制。

String到Object：视为同型，。

一般获得Envelope后：

设置值：

Variable\* var = env -> GetVariable(1);

var -> Write<int>(12);

var -> Write (data, 1024, true);

var -> Write(str, true); //非基值类型可用