ICE iPush® Communication Server

Embedded

iceHMsg.ocx

Programming Guide

By: ICE Technology Corp., March 10, 2004

Ver: 1.3.7

E-Mail \ service@icetechnology.com Tel \ +886-2-23961880 Fax \ +886-2-23961881

Table of Content

Introduction

IceHMsg is an ActiveX control that used for data representation and manipulation. It has a property dictionary whose data can be accessed with string indexes, and a text segment, both are optional. It can encode the data it represents into a byte array, which is useful to be transmitted through iPush® Embedded, and the decoding function is also provided.

This document is organized as the following sections:

- **Properties and Text** describes data types as well as data access interface
- **Producing outgoing messages** describes function used to prepare messages for transmission
- **Processing incoming messages** describes function used to decode such transmission format.

Distribution form and installing

IceHMsg is distributed as a DDL file **iceHMsg.ocx**. Before use it, you should register it as a system resource, by issuing just one command:

regsvr32 < *path to IceHMsg* >**iceHMsg.ocx**

Then you can see and use it as any ActiveX controls in your system.

Properties and Text

An **iceHMsg** object may contain many properties. A property can be a string, an integer number, a real number, or a Boolean value. These properties are stored in a dictionary data structure and accessed with their name string.

For each data type, a pair of set/get methods are provided to access a particular property. For example, to set the property **BitInput** of an iceHMsg object **msg** to True, you can use the **setBooleanProperty** method demonstrated below:

msg.setBooleanProperty "BitInput", True

and use the getBooleanProperty method to retrieve that value afterward:

BitValue = msg.getBooleanProperty("BitInput")

The following lists are those data access methods corresponding data types supported by iceHMsg:

Data Encapsulating and Extracting Method:

- **setBooleanValue (propName as String, propValue as Boolean)**

  Set the value of Boolean property named propName to propValue.

- **getBooleanValue (propName as String) As Boolean**

  Retrieve the value of Boolean property named propName.

- **setByteValue (propName as String, propValue as Integer)**

  Set the value of Byte property named propName to propValue.

  A Byte value is an integer number ranges from -2^7 to 2^7-1

- **getByteValue (propName as String) As Integer**

  Retrieve the value of Byte property named propName.

- **setShortValue (propName as String, propValue as Integer)**

  Set the value of Short property named propName to propValue.

  A Short value is an integer number ranges from -2^{16} to $2^{16}-1$.

- **getShortValue (propName as String) As Integer**

  Retrieve the value of Short property named propName.

- **setIntValue (propName as String, propValue as Long)**

  Set the value of Int property named propName to propValue.

  A Int value is an integer number ranges from -2^{31} to $2^{31}-1$.

- **getIntValue (propName as String) As Long**

  Retrieve the value of Int property named propName.

- **setFloatValue (propName as String, propValue as Single)**

  Set the value of Float property named propName to propValue.

  A Float value is a single-precision real number.

- **getFloatValue (propName as String) As Single**

  Retrieve the value of Float property named propName.

- **setDoubleValue (propName as String, propValue as Double)**

  Set the value of Double property named propName to propValue.

  A Double value is a double-precision real number.

- **getDoubleValue (propName as String) As Double**

  Retrieve the value of Double property named propName.

- **setStringValue (propName as String, propValue as String)**

  Set the value of String property named propName to propValue.

- **getStringValue (propName as String) As String**

Retrieve the value of String property named propName.

An iceHMsg object may also have one text segment, which is represented as a string. The interface to access that text segment is:

- **setText (text as String)**

  Set the text segment to text.

- **getText () As String**

  Retrieve the text segment.

There is also methods that remove properties and text segment from a iceHMsg object respectly:

- **clearProperties ()**

  Remove all properties.

- **clearText ()**

  Remove text segment.

Packing Outgoing Message

It is very simple to transform the data of an iceHMsg object into a byte array, which is useful for transmission. The interface is:

- **packSubjectMessage () As Variant**

AS THE FUNCTION NAME IMPLIES, THE RETURNED BYTE ARRAY, WHICH IS WRAPPED AS A VARIANT OBJECT, IS DESIGNED TO BE USED AS IPUSH® SUBJECT DATA, BUT NOT LIMITED TO IT.

Take an example:

```
msg.setFloatProperty “x”, 3.0
```

msg.setFloatProperty “y”, 4.0

msg.setText “point#1”

bytesMsg = msg.packSubjectMessage()

As result, bytesMsg contains the transmission form of property x set to 3.0, property y set to 4.0, and text segment “point#1”.

Parsing Incoming Message

Transform encoded data back into a iceHMsg is straightforward, the interface is:

- **parseSubjectMessage (data) as Boolean**

This method returns a Boolean value indicates the parsing is successful or not. The data type of parameter data should be byte array, just like the subject data transmitted from iPush® ActiveX API. Note that previous text segment and all properties will be removed before parsing, even the parsing fails afterward.

For example, with the previously constructed bytesMsg transmitted from iPush® as variant data:

msg.parseSubjectMessage(data)

As a result, msg contains property x, y and text segment as msg in the previous example.

floatX=msg.getFloatProperty (“x”)

and x will be 3.0

floatY=msg.getFloatProperty (“y”)

and y will be 4.0

strText = msg.getText

and we will get strText will be “point#1”

Default Accepted Data Format

Only data format that well defined in I/O module can be recognized by iPush® Embedded. Accepted data format could be expended by developer, please refer to IOModule programming guide for detail description. Following table are default data format accepted by iPush® Embedded:

Default Data Format accepted by iPush® Embedded

Property Name	Accepted Data Type	Example
AnalogInput	Float	Floating number like 1.42857 or 3.14159
AnalogOutput	Float	Floating number like 1.42857 or 3.14159
DigitalInput	Integer, Byte	0-65535, 0-255
DigitalOutput	Integer, Byte	0-65535, 0-255
BitInput	Boolean	TRUE, FALSE (Upper Case needed)
BitOutput	Boolean	TRUE, FALSE (Upper Case needed)
Report	N/A	N/A