Messaging API

Developer Guide

Contents

Introduction

Read-a-Card provides a Windows messaging based API (application programming interface) allowing other software applications to be developed that make use of Read-a-Card's functionality. Applications that register to receive Read-a-Card's notification messages will receive the card ID, card type, reader ID (where available) and timestamp, whenever a new card is presented.

Messaging Overview

To use the Read-a-Card messaging API, your application must register with Read-a-Card by sending a custom Windows message to the Read-a-Card main window. That message provides Read-a-Card with your application's window handle.

Once registered, Read-a-Card will notify your application whenever a card is presented. To do this, Read-a-Card sends a WM_COPYDATA message to your application using the window handle that was provided in the registration step. The WM_COPYDATA message is a standard method of transferring data in shared memory between two applications under Windows. The data transferred via this message contains the card ID, card type, reader ID (where available) and timestamp as 4 unicode character strings.

Because your application will be assigned a different window handle each time it runs, your application will need to send the register message to Read-a-Card whenever it or Read-a-Card is restarted.

Ideally, when your application closes or no longer requires the Read-a-Card notification messages, it should send an unregister message to Read-a-Card to stop the notification messages being sent.

Implementation Details

This messaging scheme may be implemented using any Windows application development environment that supports the Win32 API. Examples are provided in Visual C++ using MFC and in VB.NET but the steps involved are described here in more detail to allow other development environments to be used.

To register with Read-a-Card:

1. Find the Read-a-Card window handle. This can be achieved using the Win32 API FindWindow function to look for a window with the title "Read-a-Card".

2. Create a custom message ID using the Win32 API RegisterWindowMessage function with "READACARD" as the argument.

3. Use SendMessage to send a message to the Read-a-Card window. The message ID should be that created in step 2 and the LPARAM should be your application's window handle. The WPARAM is an 8 bit action mask that tells Read-a-Card what actions it should take itself when a card is presented. The details for the mask format are as follows:

   X X X X X X X X

   7 6 5 4 3 2 1 0

   Bit 7 = Not Used

   Bit 6 = Log Action Disabled 0 = On , 1 = Off

   Bit 5 = URL Action 0 = On , 1 = Off

   Bit 4 = Command Action 0 = On , 1 = Off

   Bit 3 = Sound Action 0 = On , 1 = Off

   Bit 2 = Clipboard Action 0 = On , 1 = Off

   Bit 1 = Keyboard Action 0 = On , 1 = Off

   Bit 0 = Successfully Registered 1 = On , 0 = Off

   For example, a WPARAM of integer value 1 will register with Read-a-Card and leave all functions on, while a value of 3 will register with Read-a-Card and also turn off the keyboard action.

Once registered, your application's window handler will receive WM_COPYDATA messages from Read-a-Card whenever a card is presented. The lpData member of the received COPYDATASTRUCT object can be interpreted as a pointer to a Read-a-Card structure comprising 4 null-terminated unicode character arrays each having a maximum length of 256 bytes.

The following shows the C++ structure for this data:

```
typedef struct {

      WCHAR cardID[256];

      WCHAR readerSerial[256];

      WCHAR cardType[256];

      WCHAR timeStamp[256];

} READ_A_CARD_DATA;
```


Depending on the development environment of your application, these character arrays may require appropriate conversion to be displayed as string types. See the VB.NET sample code for an example of this type of conversion.

Configuring Read-a-Card

Read-a-Card is designed to be configured by an administrator using its tab-based user interface, and run under limited user privileges which prevents the user from changing any configuration parameters. As an alternative, you can configure Read-a-Card by directly writing the file 'Read-a-Card.ini' with the required settings.

On Windows 2000/XP this is located in the folder:

```
c:\Documents and Settings\All Users\Application Data\Read-a-Card\
```

On Windows Vista, 7 and 8 this is located in the folder:

```
c:\ProgramData\Read-a-Card\
```

Technical Support

We include both the source code and executable versions of two example applications that demonstrate the use of Read-a-Card via the messaging API. These have been built using standard Microsoft compilers, and should provide enough information to guide you through creating your own integrated solution.

If you have any queries about these examples, or the API in general, please contact us via email using the address info@read-a-card.com although we may not necessarily be able to help with queries on other languages or development environments.

