Portable Vital Signs Monitors Interface Specifications

Exchanging Information with the VistA Vitals Application

October 2005

Department of Veterans Affairs
Office of Information & Technology
Office of Enterprise Development

Revision History

Date	Description	Author
6/8/2005	Created interface specifications	E. Lickerman and B. Ackerman
6/13/2005	Edited content	C. Peterson
6/27/2005	Minor content modification	C. Peterson
10/18/2005	Minor content modification	C. Peterson

This page intentionally left blank for double-sided printing.

Table of Contents

This page intentionally left blank for double-sided printing.

Introduction

The goal of this VA project is to interface small, portable vitals signs monitors with the Vitals application within VistA. These monitors are produced by many companies and yet are very similar in their functionality. All provide the same basic set of vitals measurements, though there is some expected variety in the methods they use to acquire these values.

Many of these devices are capable of interfacing with outside software systems. The first two that were examined use an RS232 port but there is no expectation that this will be the case with all such monitors. In the future there may be a need to interface with devices that use different physical interconnects. The first two devices also provide slightly different mechanisms and command sets for software interfacing. One requires commands to be sent in ASCII text directly over the serial port. The other provides a Windows Software Development Kit (SDK) that handles these low level details for the programmer and allows easier construction of Windows applications that request information from the monitor.

Rather than write a custom interface for each device used within the VA system, this specification is being provided. The goal is that a vendor will implement this specification for its own devices and that any software that meets this specification will automatically work with the Vitals package. The specification is intended to be short, simple and easy to implement as well as being platform and development language independent.

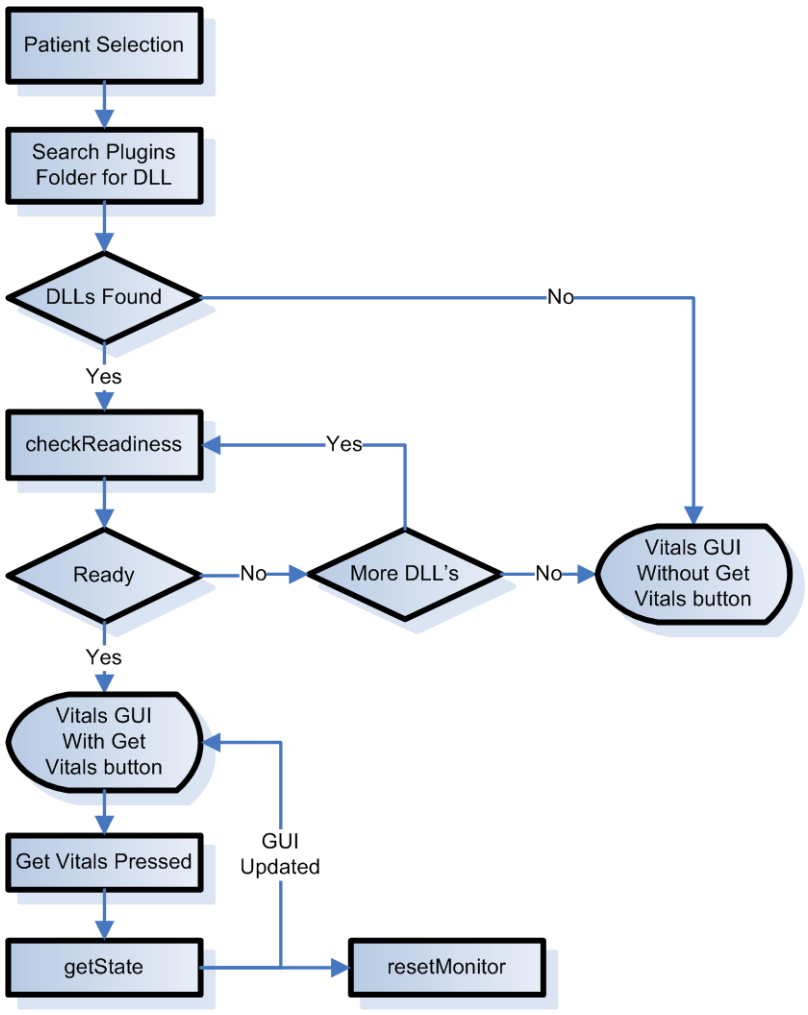
Definitions

- **DLL**: A Microsoft Windows Dynamic Link Library. A library of software functions that can be linked to anther program at runtime, as opposed to compile time. For the purposes of this project, DLL is defined as a Standard Microsoft dynamic link library – not an ActiveX DLL. It is acceptable, however, to "wrap" an ActiveX component into a Standard Microsoft DLL.
- **RS232**: A standard interface connection that provides for only modest transmission rates & is often used with modems. For the purposes of this specification, RS232 cables are taken as 9-pin cables with D-type connectors on either end and, connect to the COM port(s) on a personal computer.
- **Vendor Software**: A software program to be written by the vendor of a given small vital signs monitor, conforming to this specification and facilitating communication between the Vitals application and the given small vital signs monitor.
- **Vital Signs**: Measurements that can be taken on a patient that reflect the presence of life in that patient. The set of vital signs typically includes pulse rate, respiratory rate, blood pressure, body temperature and oxyhemoglobin saturation (O2 sat).
- **Vitals Application**: A specific software program that is used to record vital signs taken from patients. This VA application is written in M and Delphi.
- **Vitals Monitor** (also known as "monitor"): A small portable device produced by the vendor that can measure a small set of "Vital Signs" on a patient.
- **XML:** Extensible Markup Language. A flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. XML is a formal recommendation from the World Wide Web Consortium (W3C) similar to the language of today's Web pages, the Hypertext Markup Language (HTML).

Interface

The vendor shall supply a DLL that implements a simple interface as described in this document. The VA Vitals application will use the functions of the DLL supplied by the vendor. The vendor software will then interact appropriately with the monitor. The monitor may return results to the vendor software, and the vendor software will then return properly formatted results to the Vitals application.

If the Vitals application requests vital signs measurements, the vendor software will respond with a pointer to a small XML document representing the results of the request. The XML document will conform to the Document Type Definition (DTD) supplied in this specification. For other requests the vendor software will respond with the appropriate value as described in this specification. Please refer to the diagram below for typical function call order:



Commands

Due to the simplicity of the interaction required, at this point only four commands are specified that the software must accept from the Vitals application.

1. **checkReadiness**. This is the first command the Vitals software will issue to the vendor software. It is to check if the software is in a state in which it can return results. To be in a ready state, the software must be running (powered on) and able to interact with its monitor. This function must return false if the software (or firmware) version of the device is earlier than the Vitals certified version, even if it is capable of communication.
   Parameters: none
   Return type: Boolean
   Return values: True if the software is in contact with one of the vendor's monitors, False otherwise. If another vendor's monitor is connected to the PC (and not this vendor's monitor) this command must return false.

2. **getBufferLength**. The vendor will allocate a buffer that can accommodate any length of string for any XML documents that are possible output of a single instrument read. This method allows the Vitals application to discover how much memory the vendor application has allocated for this buffer. The Vitals application will call this method once at startup, so the vendor application cannot change its size after this method is called.
   Parameters: none
   Return type: 32 bit integer
   Return values: The (unchanging) size of the buffer the vendor software has allocated for XML return strings.

3. **getState**. This command is a request to retrieve the current state of the monitor. Any values for any vitals signs should be read from the monitor, expressed in the XML return format placed in the buffer, and returned to the Vitals application.
   Parameters: none
   Return type: pointer
   Return value: A pointer to the beginning of the buffer containing the string which is an XML document conforming to this specification and containing the information about the current state of the monitor, i.e. the vital signs currently in the monitor. The string is represented by a null terminated character array and is encoded as ASCII text.

4. **resetMonitor**. The Vitals application has successfully read the state of the monitor and, to avoid reading the same values a second time, requests that all the values in the monitor be reset. This function must also clean any buffers or memory used in the construction or passing of information.
   Parameters: none
   Return type: Boolean
   Return value: True if the monitor was reset; False otherwise.

DTD for Vital Signs Results

Below is the XML DTD:

```
<?xml version="1.0>
   <!ELEMENT Vitals (Vendor,Model,Serial_Number,Result*)>
   <!ELEMENT Vendor (#PCDATA)>
   <!ELEMENT Model (#PCDATA)>
   <!ELEMENT Serial_Number (#PCDATA)>
   <!ELEMENT Result (Value,Units,Time_stamp)>
   <!ATTLIST Result name(Systolic_blood_pressure|
                    Diastolic_blood_pressure|
                    Mean_arterial_pressure|
                    Oxygen_saturation|
                    Body_temperature|Pulse) #REQUIRED
                    >
   <!ELEMENT Value (#PCDATA)>
   <!ELEMENT Units (#PCDATA)>
   <!ATTLIST Units name (F|C|mmHg|BPM|%) #REQUIRED>
   <!ELEMENT Time_stamp (#PCDATA)>
   <!ATTLIST Time_stamp year CDATA #REQUIRED
                    month CDATA #REQUIRED
                    day CDATA #REQUIRED
                    hour CDATA #REQUIRED
                    minute CDATA #REQUIRED
                    second CDATA #REQUIRED
                    >
   <!ELEMENT Method (#PCDATA)>
   <!ATTLIST Method  name CDATA #REQUIRED>
```

Example Results Document

This is an example of the results document that comes from the vendor.

```
<?xml version="1.0" encoding="ASCII"?>
    <Vitals>
    <Vendor>Welch-Allyn</Vendor>
    <Model>Vitals 4000</Model>
    <Serial_Number>1122992929</Serial_Number>
    <Result name="Systolic_blood_pressure">
            <Value>120</Value>
            <Units name="mmHg"></units>
            <Time_stamp year="2004" month="7" day="30" hour="13" minute="15" second="30">
            </Time_stamp>
    </Result>
    <Result name="Diastolic_blood_pressure">
            <Value>80</Value>
            <Units name="mmHg"></units>
            <Time_stamp year="2004" month="7" day="30" hour="13" minute="15" second="30">
            </Time_stamp>
    </Result>
    <Result name="Mean_arterial_pressure">
            <Value>103</Value>
            <Units name="mmHg"></units>
            <Time_stamp year="2004" month="7" day="30" hour="13" minute="15" second="30">
            </Time_stamp>
    </Result>
    <Result name="Oxygen_saturation">
            <Value>95</Value>
            <Units name="%"></units>
            <Time_stamp year="2004" month="7" day="30" hour="13" minute="15" second="30">
            </Time_stamp>
    </Result>
    <Result name="Body_temperature">
            <Value>98.6</Value>
            <Units name="F"></units>
            <Method>predicted</Method>
            <Time_stamp year="2004" month="7" day="30" hour="13" minute="15" second="30">
            </Time_stamp>
    </Result>
    <Result name="Pulse">
            <Value>84</Value>
            <units name="BPM"></units>
            <Method>pulse_ox</Method>
            <Time_stamp year="2004" month="7" day="30" hour="13" minute="15" second="30">
            </Time_stamp>
    </Result>
    </Vitals>
```

Requirements

XML Document Tags and Requirements

1. **Spacing:** The spacing used in the example results document is to be followed exactly paying particular attention to the spacing within the Time_stamp tag.

2. **Capitalization:** The capitalization used in the example results document is to be followed exactly.

3. **Empty Results:** The XML document shall not include any result blocks for readings that were not measured. If, for example, temperature, SpO2, and pulse are taken but not blood pressure, the two results blocks pertaining to blood pressure are to be omitted altogether as opposed to sending a value of 0 or an empty tag.

4. **Vendor tag**. This contains the name of the vendor as PCDATA. The vendor shall decide what to put for name. Once decided it should never be changed without notice to the VA. This name is case sensitive.

5. **Model tag**. PCDATA within this tag indicates the model name or number of the monitor. This is set at the vendor's discretion.

6. **Serial Number tag**. PCDATA within this tag indicates the serial number assigned to the monitor by the vendor. This should uniquely identify the device.

7. **Result tag**. This is the parent tag of all elements that, taken together, describe the result of a single vital sign measurement.

8. **name attribute**. Contains CDATA indicating the name of the vital sign. These names are set and defined by this specification, i.e. vendor specific names will not be passed in this attribute. The vendor is responsible for deciding the appropriate translation from its internal naming scheme to the naming scheme described in this document.

9. **Value tag**. This contains PCDATA representing the value of the measurement. It should be text for the decimal number of the value. None of these measurements require scientific notation and it is not intended that the program encode the binary value of the number in any way. It should be represented as human readable text for the number in Arabic numerals. It should follow the convention that a "." separates the whole number part from the decimal fraction. No commas (",") should be used to separate digits in the whole number portion (e.g. 10000 not 10,000). The number of digits after the decimal point should reflect the precision of the monitor for the given measurement, i.e. this specification does not limit this.

10. **Units tag**. This contains PCDATA indicating the units that apply to the value of the measurement. The DTD specifies these units and if the monitor uses different units the

vendor software is responsible for translating them to the units allowed by the DTD. Acceptable units are:

a. F stands for "Fahrenheit," the most common unit of measure for body temperature.
b. C stands for "Celsius" or "Centigrade," the metric unit for body temperature.
c. mmHg stands for "millimeters of mercury" and is a unit of pressure commonly used to report blood pressure.
d. BPM stands for "Beats per Minute" and is the most common way of reporting the pulse or heart rate.
e. % in this system is used to report the Oxygen saturation of the blood.

11. **Method tag**. This contains PCDATA that represents the method used to produce the result. It is vendor specific and a method code and name may combine several concepts. Examples:
    a. The predicted vs. the actual measured temperature. Many vitals monitors can extrapolate the body temperature from the temperature change in the temperature probe, before the probe temperature has equalized with that of the surrounding body.
    b. Pulse can be taken from the pulse oximetry finger clip or from the blood pressure cuff.

12. **Time_stamp tag**. This has attributes that indicate the time the parent result was produced by the monitor, using the monitor's clock which is not guaranteed to be synchronized with the PC's system time. Each attribute contains CDATA that indicates the number of years, months, days etc. as text. The numbers will be base ten integers expressed in standard Arabic numerals, i.e. not hexadecimal.

Coding Requirements

The vendor application is not responsible for supplying the Vitals application with standard codes such as LOINC or SNOMED. The vendor shall supply the VA with mappings of their result types to standard codes. The combination of the result name, the result method and the vendor should be sufficient to code the result. The Vitals application will be responsible for using or not using this information.

Hardware Support Requirements

A Pentium Class system capable of running Microsoft Windows XP (or greater) with available RS232 (COM) port is the only hardware requirement.

Installer Requirements

The vendor will supply a graphical installer for the given PC platform. This installer must be easy to use. If the user accepts the default settings, it should result in a successful install under most circumstances.

1. The installer should also be capable of updating an existing installation to a new version.
2. The installer should be a "single" installer that consists of a single executable.
3. The installer shall install the DLL responsible for communication into the **/program files/VistA/Vitals/Plugins/** folder.
4. The installer shall install all other supporting files needed by the above DLL into the **/program files/<vendor name>/** folder. The vendor name used here must be consistent throughout all subsequent installations.
5. The installer shall NOT install any extraneous files nor overwrite any critical system files.

Failure Mode Requirements

1. Vendor software loses connection with vitals monitor.
   a. The software should attempt to re-establish connection.
2. Vendor software crashes.
   a. The rest of the system should be unaffected by a vendor software crash.
   b. The Vitals software should be able to attempt a re-start of the vendor software.
3. The Vitals application crashes.
   a. When re-started the Vitals application is responsible for requesting a re-establishment of the socket between it and the vendor application. The vendor software shall support this. The unexpected termination of the connection to the Vitals application should be handled well. The vendor software must not crash under such circumstances and be immediately available for new connection attempts by the Vitals software.

General Requirements

The vendor software must flawlessly identify its own monitors and never attempt to acquire or return results read from another vendor's monitor.

Multi-Threading

It is recognized that the construction of the return XML will take a finite time. The interface must be a single-threaded interface where each request will be processed and returned in the same order that is was transmitted.

Instrument Operating Modes

1. **Request-Response Mode**: This is the default operating mode for the software. In this mode the vendor software waits for a request from the Vitals software before sending a response. This involves the vendor software listening on the socket for incoming text commands and responding to valid ones.
2. **Monitor mode**: When the instrument is set to monitor mode it will take readings at intervals and send them to the software. This mode is not supported by the Vitals application and therefore not supported by this specification.