Working Group on
Neurocritical Care Informatics

Recommendation

Medical Device Connectivity

Document Number:	R-001
Date:	June 29, 2015
Updated:	September 1, 2016
Status:	**DRAFT**

Participants

This Recommendation has been developed by the Working Group on Neurocritical Care Informatics. Contributors to this Recommendation are as follows:

Primary Author: Anna Rodriguez, PhD, Moberg Research, Inc.
.

Participants
Dick Moberg, Moberg Research, Inc
Matt Ramsey, Moberg Research, Inc.
Gary Trapuzzano, Moberg Research, Inc.

Others
Michael Schmidt, PhD, Columbia University

Contents

Purpose

This document is intended to provide guidance to Medical Device manufacturers desiring to design a communication protocol that allows external systems (data aggregators, repositories, etc.) to acquire data from their device. This document describes the need for Medical Device connectivity as well as the problems encountered with existing Medical Device communications.

Scope

The scope of the document is limited to that of neurocritical care; however, the recommendations are deemed to have value across other disciplines where Medical Device connectivity plays a role in patient care.

Terms and Definitions

Medical Device
Refers to a device that is a generator of data, which may be physiologic or non-physiologic.

Data Receiver
Refers to a system that receives data from one or more Medical Devices.

Medical Device Connectivity
Refers to the means by which a Medical Device provides information to another system which could be a Medical Device or other receiver of the information.

Medical Device Interoperability
Refers to the two-way communications between Medical Devices with the purpose of controlling one or both of the devices. Interoperability is not addressed in this Recommendation as there are few Medical Devices that permit their control by a generalized external source.

Clinical Need

Neurocritical care is a medical discipline that deals with complex neurosurgical, neurological and medical problems in critically ill patients who have suffered acute life-threatening brain injuries, including traumatic brain injury, stroke (e.g., ischemic, hemorrhagic), encephalitis, or status epilepticus.

In many neuroscience intensive care units (neuro-ICUs), multimodal monitoring is routinely performed and has become a standard of care. Multimodal monitoring is defined as the continuous, simultaneous evaluation of cerebral function and vital signs from multiple modalities in a single patient, with the goal of avoiding or mitigating secondary brain insults [1-4].

The increase in the number of monitored modalities has led to the creation of a data-rich environment, but one in which information extraction and consolidated review that results in enhanced situational awareness of patient condition and improvements in care quality is still suboptimal [5, 6]. The data collected from each patient still varies from hospital to hospital and is rarely collected in a standardized format. A significant limitation to the use of advanced informatics in current neuro-ICUs is the highly proprietary nature of the data collected by the multiple stand-alone devices. Whereas in consumer electronics an interoperable, “plug-and-play” approach is expected from all devices, this is not true in the neurocritical care environment [5, 7] and in healthcare in general [8, 9]: still numerous barriers exist to the integration of data collected by multiple sources. This results in an increase in the possibility of adverse events, reduced clinician productivity and in the missed added diagnostic value that would be provided by integrated information [8, 9]. Up to this day, powerful informatics tools, such as clinical decision support and multi-parametric alarms, are only used in institutions that have developed their own unique data infrastructure and these tools are rarely scalable or adaptable for routine care. As an added challenge in neurocritical care, multimodal monitoring and diagnosis cannot solely rely on the data collected by electronic medical records: in order to provide timely and effective care, high-resolution and well-annotated data streams need to be utilized, as opposed to the information collected at sparser intervals for inclusion in the medical records [1, 5, 10].

Historical Note

Device integration is commonplace in the computer world, but has not happened in neurocritical care, a field in which data is so critical to patient management. The reason for this lack of integration is multifaceted and has been influenced by culture, corporations, and governmental policy and regulations. The problem is compounded by a lack of sufficient commercial and regulatory incentives for integration.

Historically, data outputs from medical monitors were utilized primarily by researchers, not clinicians, and were not widely requested. As a result, the quality and robustness of data communication protocols is extremely variable, if available at all. Even with the increasing demand for such communication, regulatory barriers and economics tend to minimize changes a manufacturer makes to Medical Devices once released, thus slowing any trend toward improvement. Additionally, larger device companies may perceive a standardized device interface as a risk in that they may lose sales to competing connected products.

State of the Art

Today there is a standard for Medical Device communications that has been vetted over the past 20 years but only adopted by a few vendors. This is the ISO/IEEE 11073 family of standards. However, without widespread adoption, the standard has become obsolete in some areas and has not kept up with the needs of modern Medical Devices. Several groups are working on updates to the standard. These are the Center for Medical Interoperability [11] and the Medical Device “Plug and Play” Interoperability Program [12].

Several factors may drive the further development and adoption of a standard. The Department of Defense in the U.S. is investigating an Autonomous Critical Care platform in which there will be a tight coupling of devices [13]. Several organizations have been funded to develop systems of connected devices that adopt a standard protocol.

In the U.S. the HITECH Act of 2009 provides incentives for Medical Devices to be connected to the medical record; however, the dominant medical record companies have not promoted a Medical Device standard, just a means to connect to whatever protocol the device uses. An example is Cerner's iBus [14]. In addition, medical records usually require only sparse data from medical devise (every few minutes to every hour) as opposed to higher resolution data that devices record.

Interconnected systems have long been uncharted territory in terms of regulations and lines of authority. In 2016, the U.S. Food and Drug Administration published a draft guidance for interoperable medical devices [15]. The document highlights design considerations, including security and risk management, that are unique to interconnected systems and it offers guidance for verification and validation.

Recommendation 1: Use of Standards

It is recommended that manufacturers utilize standard communication protocols when applicable and when available.

It is realized that there exists no widely adopted Medical Device communication standard at the time of writing this Recommendation. However, manufacturers are encouraged to follow the work of organizations helping to develop such a standard. In the absence of a communications standard, the remaining recommendations in this document should be followed.

Recommendation 2: Digital vs. Analog Communications

The use of analog data communications is discouraged unless some method of positive signal identification is utilized.

Analog data outputs consist of a single voltage that corresponds to an output of a Medical Device (generally to a physiological measurement). As opposed to digital methods, analog communications contain no information identifying the source of that signal, error conditions, etc. In multimodal monitoring it is common to record measurements from more than one device. Since analog signals cannot inherently identify themselves, there is the potential for errors when connecting more than one Medical Device using analog communications to a Data Receiver.

Recommendation 3: Desirable Protocol Information Content

It is recommended that manufacturers ensure the following content is transmitted from the Medical Device.

3.1 Device Identification

The device needs to be uniquely identified with the name, model number, manufacturer, and, if possible, the serial number. The recommendation is to use the Unique Device Identifier (UDI) format that is being adopted globally for device identification [16].

Note: Plug-and-play of medical devices requires a positive identification of devices early in the device-receiver transaction process.

3.2 Protocol version identification

An indication of the version of the communications protocol should be provided. It is important for the Data Receiver to ensure that the communications protocol is the one that is expected from (and compatible with) the device identified. This identifier should only be changed when a new version of the protocol software would cause an error in reading the data by the Data Receiver. Backwards compatibility with prior versions of the protocol is strongly recommended.

Note: Manufacturers have been known to change the communications protocol with no notification to users of the device or to developers of software that receives data from that device. This identification will prevent errors associated with incompatible software versions.

3.3 Patient Identification

It is recommended that an identifier for the patient be transmitted to the Data Receiver if this information is entered into the Medical Device.

3.4 Data

It is recommended that all clinically useful data (visible to or used by the clinician) be transmitted to the Data Receiver. The following should be transmitted:

1. Data Label: For example: ICP, PbtO2, etc. It is recommended that a standard nomenclature be followed.
2. Data Value: The value of the data point. For example: 27. The protocol should not change the way the values are represented. (For example, one product examined changes to scientific notation for very low values and this is undocumented in the device communication specifications.)
3. Data Units: For example: mmHg. It is recommended that one type of unit be used for the communications protocol in cases when the user can change the units on the Medical Device (For example, if a display can show Fahrenheit or Centigrade, select one of these as the units of the communications output as they can be easily converted).

Any data quality metrics detected by or computed by the Medical Device should be transmitted in a similar format.

3.5 Events

It is recommended that all user-marked or automatically-marked events that appear on the Medical Device's display be transmitted via the protocol.

3.6 Alarm Conditions Signaled by the Medical Device

It is recommended that all alarm conditions be transmitted to the Data Receiver.

> Note: Often this sort of information will be represented by a data type / alarm state / alarm threshold triplet, allowing the alarm declaration algorithm to remain a "black box". For instance, a device might declare an alarm for measurement X when 3 of its last 5 measured values have been above a threshold Y. The data type / state information adequately captures the history of an alarm's assertion while the threshold value provides contextual information of the system's configuration at the time that the alarm was declared.

3.7 Device System Status

It is recommended that device system status messages be transmitted. As opposed to alarms, these are messages regarding the status of the system, such as disconnected sensors, incompatible sensors, computer/memory problems, etc. In many systems these are called INOPS (for "inoperative") to distinguish them from alarms.

> Note: An example of the use of system status is knowing that an SpO2 sensor is not connected. This information can be used to explain the absence of SpO2 in the data record.

Recommendation 4: Message Structure

4.1 Required Attributes of a Message Structure

Every communication protocol must employ some form of a message structure that allows receivers of those messages to determine where one message ends and the next message begins.

- Deterministic organization
  The structure of a message must be able to be described by a set of rules, such that anyone following those rules is able to correctly interpret the message content without any possibility of error or ambiguity.

  > Note: For example, if a device is capable of measuring X, Y and Z but at certain times it only outputs X and Y while at other times it outputs Y and Z, the structure of the data message must be such that there is no ambiguity between those output

cases. For this particular example, a message format that transmits measurements as data type id / value pairs would allow receivers to differentiate between the two cases outlined above. An alternative way of dealing with this situation would be to use a message structure where X, Y and Z were always represented, but special “sentinel” values are passed to signal when a valid value is not available.

- Units of transmitted data
  The units must be identified in the data messages.
- Labels of transmitted data
  The label should be contained in the data messages.

4.2 Desirable Attributes of a Message Structure

- Identifiable start of message sequence
- Message type identifiers
- Message length
- Timestamp or Sampling sequence number
  When timestamps are used, they should represent the time at which the data points were obtained rather than their time of transmission. Thus, if a device collected multiple data items at a single point in time but transmitted those values in separate messages, each of those messages should have the same timestamp.
- Message sequence number
  This allows receivers to detect when one or more messages have been lost. Generally an 8 or 16 bit counter that periodically rolls over would be sufficient for this purpose.
- Message checksum
- Numeric data bundling.
  If a system periodically produces a set of data values for a set of data items, passing all of those values together in a single message allows the client to easily keep those values synchronized with each other. An alternative approach might be to employ multiple messages tagged with a common sampling sequence number or timestamp.

4.3 Undesirable Attributes within a Message Structure

- Language dependent content
  If a source device can be configured to operate in more than one language, proper interpretation of it is message content should not depend on clients being able to recognize all of the different parameter labels that may be associated with values.

  Note: For example, if ECG and EKG are two different labels that might be applied to the same parameter depending on the language configuration, the message structure should not use the label string to identify the parameter. A better approach would be to assign an identifier number for that parameter and potentially pass both value and current label along with the parameter identifier. Such an implementation would provide the maximum flexibility to downstream clients.

Recommendation 5: Design Considerations – Needs Assessment

It is recommended that a needs assessment be conducted in the initial phase of designing a Medical Device communications protocol. The following paragraph provides some topics to be considered in this assessment.

The first step of designing a successful interface protocol is to try to identify all of the needs to be addressed by that interface. Does the interface need to be usable by client systems without any special receiving software? If so, an ASCII interface protocol whose output could be captured to a text file using universally available applications such as HyperTerminal could be an option. Does your system's data need to be accessible to more than one client at a time? If so, your system may need to support those clients through multiple physical interfaces (serial or USB ports) or through a shared physical interface such as an Ethernet port. What resolution will be sufficient for representing data values in your output? For example, if you compute a numeric measurement to higher precision than you display it on your device, what precision do you want to employ in your output protocol? Higher precision values may be more useful for researchers while posing conceptual problems for clinical users of that interface when received data is in apparent disagreement with data displayed by your device. What are potential future uses of the system and the collected data? By considering potential advancements in the field and promising research uses, the interface protocol can be designed to be flexible and meet a wide array of needs.

Recommendation 6: Design Considerations – Transport Protocol Selection

It is recommended that a transport protocol be chosen according to the needs identified.

The first decision point is typically to choose what mode(s) of transport will be utilized by the interface protocol. For existing systems, the choices will be limited to the hardware already present (and available) in the system. For systems under design, careful consideration should be given to the types of communication hardware to be included. Key factors to look at include the bandwidth required for the available data, the number of desired connection points, positive association of the data to the device that is generating it, and electrical isolation.

There are three leading modes of transport:

1. Serial (RS-232)

RS-232 serial communications provides a point-to-point connection, which makes the pathway between the host device and the client clearly identifiable, but limits each serial port to serving only one client.
Bandwidth of RS-232 is fairly limited. While capable of being pushed to over 1Mbit/s, most standard devices max out at 115Kbit/s and care must be taken with maximum cable length and driver circuits to maintain reliable communications as data rates get into hundreds of Kbit/s.
RS-232 is relatively easy to isolate, but in its standard implementation, ground is shared between the two systems, setting up a path for leakage currents between devices that can violate patient safety standards.

RS-232 serial communications is an older technology and, while it is still well supported in the embedded market, it is starting to fade from the consumer market. Desktop PCs generally still offer serial ports, although many of them now only provide an internal connector on the motherboard. Serial ports are rare in laptops and nonexistent in tablet PCs, although USB to RS-232 converters are readily available to provide the interface if it is not natively supported.

RS-232 Serial Protocol Attributes

When a protocol uses RS-232, the following settings must be specified:

Baud Rate

An adequate baud rate is required. Unless a particularly noisy operating environment is expected, higher baud rates (out of the set of standard supported rates) are often preferable to lower rates because the increased capacity allows for future growth. Offering a selectable baud rate on the source device may allow greater compatibility with clients whose baud rate is less configurable, but frequently this merely leads to increased opportunities for misconfiguration.

Data Bits

If you are implementing a binary protocol, this will always be 8. With ASCII protocols, choose 8 bits with no parity, or 7 data bits with odd or even parity.

Stop Bits

A single stop bit is almost universally used.

Parity

Good protocol message structures will incorporate a checksum as part of the message content. If you follow that path, you will not need to enable parity nor deal with the cases where the serial port driver discards received bytes with parity errors.

Flow Control

Implementing flow control often presents as many problems as it solves. It requires source devices to have a strategy for dealing with the case where the client is not ready for more data and may not signal readiness for quite some time. It also requires client devices to prepare for the possibility that enabling flow again may result in a torrent of pent up messages whose original transmission time may be impossible to determine. It is often better to employ a more rigid message structure including a checksum that readily allows clients to discard partially received messages. In most cases, the buffering provided by the serial port hardware and at the driver level will protect against the case where client applications are temporarily precluded from running in a multi-tasking system. If you must use flow control, RTS / CTS is probably the best choice since it is "sticky", whereas software flow control (XON / XOFF) requires that the other end is actually listening when the flow control characters are sent.

2. Ethernet

Ethernet is a ubiquitous technology whose capabilities have continued to expand, allowing it to thus far survive all attempts to replace it with newer technologies. It has high bandwidth - PCs generally offer Gbit/s speeds and low cost embedded processors with 100Mbit/s interfaces are readily available. Ethernet is also inherently isolated (in the standard twisted pair cable implementation), with 1500V of isolation required in the IEEE802.3 specification.

While the twisted pair cable is a point-to-point connection, the standardized protocols such as TCP/IP that run on top of Ethernet and the proliferation of hubs and switches mean that an Ethernet port is able to serve multiple clients via a single point of connection. That advantage also comes with a cost. Ethernet enabled devices will need to support an IP address, either statically assigned by the source device (presumably through a user interface) or dynamically assigned by the network. In addition, since this creates the possibility that the client device will connect to the host device over a network which may span multiple rooms where conceivably multiple client and multiple host devices may be connected, clients will face the added burden of being able to determine that they are communicating with the desired instance of the source device. Such schemes can only succeed when a unique identifying attribute can be determined that would allow such a positive association to be made. For instance, if a source device allows users to identify the patient(s) they are connected to by means of fields such as medical record number and patient name, clients can verify that they have indeed established a connection to the desired source device when provided that information. This association problem can be avoided by requiring the device to be connected to an isolated sub network limited to a single room, but this requirement is likely to run counter to customers' expectations that an Ethernet device can be plugged in anywhere on the local network. The issue of a proper host / client association over a network is a complicated proposition and ought to be considered carefully before choosing the Ethernet mode of communication.

3. USB

USB is another widely supported interface technology, especially in the consumer market, but it comes with its own set of complications. Bandwidth is good, with simple devices supporting 12Mbit/s and more advanced ones supporting 480Mbit/s. Like RS-232, USB shares ground between the two connected devices, setting up leakage current paths. Because the data signals are bidirectional and the USB interface provides power, however, isolating a USB port is more complicated than isolating a serial port.

The primary complication of using USB communication is the host/device nature of USB. The host initiates and controls all communication with the device and hosts can only talk to devices. Host to host or device-to-device communications are not supported. Higher power systems such as PCs tend to support only USB host operation, while USB devices tend to be simpler embedded systems. Likewise operating system support is much greater for host mode than for device mode (Windows does not offer any support for USB device operation). There is USB OTG (on the go) mode that supports switching between host and device mode depending on what is plugged in, but again, that is supported on a certain class of device and not available on others. Systems using data from a Medical Device are likely to be PC based, requiring the Medical Device to operate as a USB device, but the computing platform and/or operating system of the Medical Device may not support USB device operation.

The second complication is choosing the protocol to implement on top of the USB interface. There are a number of defined protocol options, as well as the ability to create your own. The protocol implementation is built into the USB device, but requires a driver to be loaded on the USB host. Of the defined protocols (referred to as classes) the following protocols have generic drivers available in most OS's and are probably the best fit for Medical Device communication:

USB Serial
The USB serial protocol creates a virtual serial port using the USB hardware. This has the advantage of supporting USB data rates but looking like a standard serial port to the USB host. One thing to be wary of is that the naming of the virtual serial port on the host can be dynamic, which makes it tricky to ensure that the program on the host system is talking to the correct device.

USB Ethernet
Similar to USB Serial, USB Ethernet creates a virtual Ethernet connection over the USB interface. While this does not require the use of a common Ethernet protocol such as TCP/IP over that connection, it will certainly be the path of least resistance. This brings some of the problems associated with the Ethernet interface above (i.e. IP address assignment), although it does remain a point to point connection, avoiding the problems with host/client associate.

Recommendation 7: Design Considerations –Protocol Format

It is recommended that a protocol format be chosen according to the needs identified.

There are three common protocol formats:

ASCII

When a protocol uses an ASCII format, the messages of that protocol tend to have the advantage of being human readable when captured to a file or routed to a terminal. ASCII protocols are more prevalent when a serial transport is being employed, but are certainly usable regardless of the transport method. HL7 is a prime example of an Ethernet based communications protocol that employs ASCII message content.

Desirable ASCII protocol attributes:

a. Protocol messages are terminated with a line feed / carriage return pair. Messages employing this structure will be clearly delineated when received by a terminal or stored in a text file.

b. Protocol messages should only employ viewable ASCII characters within their body.

c. When an RS-232 pathway is used with an ASCII format (either directly or indirectly via USB), employing parity provides an added level of confidence that the message was properly received. However, since an even number of bit errors can still pass a parity check, the use of parity should not be construed as an adequate substitute for some form of message checksum.

XML

XML based message structures have some advantages. Standard XML parsing tools can be employed to interpret received messages. When thoughtfully executed, new information elements can be added to XML messages without interfering with the ability of older receiving software to properly interpret older unaffected elements, thus ensuring a certain amount of backwards compatibility. However, XML based messages do not provide an easy path to spreadsheet applications.

Binary

Binary protocols have several advantages over ASCII protocols. One fundamental advantage is the ability to pass the same amount of information using significantly fewer bytes. This can be especially important for devices capable of producing data at a high rate, such as waveforms. Binary protocols also tend to employ more deterministic message structures.

Recommendation 8: Design Considerations – Unidirectional vs. Bidirectional Communication Protocols

It is recommended that the needs assessment determine the directional nature of the communications protocol.

The following text provides guidance in selecting either a unidirectional or bidirectional protocol.

Unidirectional

For simple devices with a limited amount of information to transfer, a unidirectional communication protocol may suffice. When a device implements a unidirectional (i.e. output only) protocol, clients are relegated to a "listen only" mode of operation. Because the source device does not listen for messages from clients, it has no way of knowing when clients are listening and when they are not. Therefore, any information required to properly interpret received messages should be present in each message. The following example illustrates this need:

> A patient monitor uses a unidirectional ASCII protocol to send out its data values. The output protocol was originally intended to feed a printer device and consequently employs a "paged" output. Each page begins with a couple of header lines to describe the data values to follow and subsequent data lines contain data values organized into columns under those headings. At 5 seconds per data line and 60 data lines per page, each page represents a 5 minute time slice. If an intelligent receiver is connected to that output port after the header lines have already been transmitted, it has no way of knowing what measurements those data values correspond to. That receiver would be obligated to throw away any data received until such time as a new set of header lines was received that would permit it to properly interpret subsequent data lines. Therefore, after being connected it could take up to five minutes before data values could be recorded from that monitor.

Bidirectional

Bidirectional communication protocols impose more arduous programming requirements on both the source device and receiver's parts but open the door to a much more robust and flexible interface. For example, if a source device can respond to a receiver's requests for device and patient information, that information does not have to be continuously transmitted as part of the source device's normal output. One situation where a bidirectional communication protocol is almost mandatory is when a source device's set of available data exceeds the transmission limits of the chosen transport method. In such cases it is recommended that the source device's communication protocol support "subscription" requests from the receiver so that it can specify the data items that it is interested in receiving.

Bidirectional protocols that require clients to request each new data set individually are generally problematic. If the client submits requests too slowly, some data sets may end up being skipped. If requests are made too often, data sets may be spuriously repeated. It is preferable for clients to be able to request the source device to transmit data as it is produced until a stop transmission request is received. Other variants of this strategy require clients to periodically refresh their request their request or to acknowledge received messages so that source devices can intelligently go quiet when the client is no longer listening.

Recommendation 9: Documentation

In order for a communications protocol to be successfully used, it must be thoroughly documented including all possible messages.

Recommendation 10: Validation

Generally speaking, when a third party needs to develop the client side for a device's interface protocol it is very helpful to have tools available to support that development. The most useful tool of all is a simulation application that can run on a standard platform such as a Windows PC. Such a simulation application should:

a. Be able to generate all possible messages that may be generated by the source device.
b. Be able to properly respond to any incoming messages that a client can send (Bi-directional protocols only).
c. Employ representative timing of data transmissions.
d. Respond to any control signals as described in the protocol documentation.

References

[1] P. Le Roux, D. K. Menon, G. Citerio, P. Vespa, M. K. Bader, G. M. Brophy, *et al.*, "Consensus Summary Statement of the International Multidisciplinary Consensus Conference on Multimodality Monitoring in Neurocritical Care," *Intensive Care Medicine,* vol. 40, pp. 1189-1209, 2014.

[2] G. Citerio, M. Oddo, and F. S. Taccone, "Recommendations for the use of multimodal monitoring in the neurointensive care unit," *Curr Opin Crit Care,* vol. 21, pp. 113-9, Apr 2015.

[3] M. Oddo, F. Villa, and G. Citerio, "Brain multimodality monitoring," *Current Opinion in Critical Care,* vol. 18, pp. 111-118, 2012.

[4] B. F. E. Feyen, S. Sener, P. G. Jorens, T. Menovsky, and A. I. R. Maas, "Neuromonitoring in Traumatic Brain Injury," *Minerva anestesiologica,* vol. 78, pp. 949-58, 2012.

[5] J. M. Schmidt, M. De Georgia, and M. Participants in the International Multidisciplinary Consensus Conference on Multimodality, "Multimodality monitoring: informatics, integration data display and analysis," *Neurocrit Care,* vol. 21 Suppl 2, pp. S229-38, Dec 2014.

[6] N. Adhikari and S. E. Lapinsky, "Medical informatics in the intensive care unit: overview of technology assessment," *J Crit Care,* vol. 18, pp. 41-7, Mar 2003.

[7] A. Sivaganesan, G. T. Manley, and M. C. Huang, "Informatics for neurocritical care: challenges and opportunities," *Neurocrit Care,* vol. 20, pp. 132-41, Feb 2014.

[8] West Health Institute, "The value of medical device interoperabilty," ed, 2013.

[9] West Health Institute, "Missed connections: A nurses survey on interoperability and improved patient care," ed, 2015.

[10] D. F. Signorini, I. R. Piper, P. A. Jones, and T. P. Howells, "Importance of textual data in multimodality monitoring," *Crit Care Med,* vol. 25, pp. 2048-50, Dec 1997.

[11] *Center for Medical Interoperability*. Available: http://medicalinteroperability.org/

[12] *Medical Device "Plug-and-Play" Interoperability Program*. Available: http://www.mdpnp.org/

[13] C. Santee. (2014) Tomorrow's Tech: The Automated Critical Care System *Future Force*. Available: http://futureforce.navylive.dodlive.mil/2014/09/tomorrows-tech-the-automated-critical-care-system-web-exclusive/

[14] *Cerner*. Available: http://www.cerner.com/

[15] FDA, "Design Considerations and Pre-market Submission Recommendations for Interoperable Medical Devices - Draft Guidance for Industry and Food and Drug Administration Staff," ed, 2016.

[16] *Unique Device Identification (UDI)*. Available: http://www.gs1.org/healthcare/udi