
Telemedicine Interoperability Alliance

**Version 0.9**

# Telemedicine System Interoperability Architecture

## Concept Description and Architecture Overview

Editor: Rick Craft / Sandia National Labs, PO Box 5800, Albuquerque, NM, 87185-0839, USA, Phone +1 505 844 8873, Fax +1 505 284-4778

**NOTICE:** Preparation of this draft was funded by the U.S. Army Telemedicine and Advanced Technologies Research Center and executed by Sandia National Laboratories. Sandia National Laboratories is operated for the United States Department of Energy by Sandia Corporation. 

## Request for Comments on:
### Telemedicine System Interoperability Architecture: Architecture Overview

Comments may be received in any format, but preference is shown for comments to be in a Microsoft Office electronic document format to facilitate compiling the comments and responses to the comments. The use of the format shown below is appreciated but commenters are also welcome to contact the editor by phone.

Comments should be separated and numbered and preferably grouped:

The comments should contain and be structured according to the following:

**Author:**

1. Author identification
   Author identification:
   Name, organization, address, fax and phone number, email
2. Author represents: self or others
   If others: name, organization, address, fax and phone number, email

Comment:

3. Comment text:
4. Reference to Working Document: document name, version, page, line number(s)
5. Reference to other documents: document name, version, chapter, page, line numbers(s)
6. Classification of each comment as one of the following:

| | | |
|---|---|---|
| **(T)** | **Typo** | Technical, grammatical, or typographical error |
| **(Q)** | **Query** | Uncertain of meaning or lack of clarity |
| **(S)** | **Suggest** | Suggestion for improvement wich does not arise from any disagreement with the approach |
| **(m)** | **Minor** | Minor disagreement with the approach taken or the terms of the standard. This indicates something which one feels needs discussion but which one does not necessarily oppose. |
| **(M)** | **Major** | Serious disagreement with the approach taken or the terms of the standard. This indicates that failure to accept the change might lead to opposing the standard in principle. In this case, reasons and useful discussion must be given as well as possible alternative solutions |

7. When making comments one should consider using examples of the problems one foresees, provide accurate references to other standards, or other documents which illustrate a point (please enclose applicable portions of a referenced document if it is not readily available).
8. In addition to comments on the current text, please suggest what kinds of changes you would like to see to accommodate your comments, with inclusion of revised phrasing where ever possible

## Comments on:
### Telemedicine System Interoperability Architecture: Architecture Overview

Please send comments to:

telemed@sandia.gov or

Rick Craft
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0839

Comments from:

Name:

Address:

E-mail:

Author represents self or other (please specify organization):

### Classification of Comments

| | | |
|---|---|---|
| **(T)** | **Typo** | Technical, grammatical, or typographical error |
| **(Q)** | **Query** | Uncertain of meaning or lack of clarity |
| **(S)** | **Suggest** | Suggestion for improvement that does not arise from any disagreement with the approach |
| **(m)** | **Minor** | Minor disagreement with the approach taken or the terms of the standard. This indicates something which one feels needs discussion but which one does not necessarily oppose. |
| **(M)** | **Major** | Serious disagreement with the approach taken or the terms of the standard. This indicates that failure to accept the change might lead to opposing the standard in principle. |
| **(U)** | **Unclassified** | Comments not falling in other categories. |

| Comment Number. | Page Number | Line Number | Comment Type | Comment |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Table of Contents

# List of Figures

# 1 Introduction

In the opening page of their book, Telemedicine and the Reinvention of Healthcare, Marc Ringel and Jeffrey Bauer state:

> Telemedicine, one of the major forces shaping the future of healthcare, is widely misunderstood. Its long-term impact on healthcare is obscured by excessive concerns with short-term policy problems, a misleading focus on narrow definitions, or utopian expectations of technology. People who overreact to telemedicine's early difficulties or underestimate its scope will be surprised by its real power. Telemedicine will ultimately revolutionize healthcare – restructuring virtually every relationship and activity that define late twentieth century medicine."

While they are very likely correct in their assertions, telemedicine is still a fledgling industry. Before Ringel and Bauer's prophecy can be fulfilled there is much clinical research and technical work left to be done.

Today, a lot of the technical work done in the telemedicine industry entails vendors integrating suites of components to create turnkey solutions for specific clinical settings ("Our system allows you to regularly monitor a patient's vital signs and glucose levels without you having to see them in person." "Our system let's you capture a series of images and send them off to a specialist for review."). While some of the systems that have been produced in this way have achieved stunning clinical successes, the industry has not yet "arrived" – there is more that can be done from a technical viewpoint that is worth pursuing.

Today, most systems are designed as turnkey solutions. Typically, it is difficult to get systems developed by different manufacturers to exchange data with other. Company A's caregiver station can't control the devices on Company B's patient station and can't retrieve clinical data from it. Similarly, telemedicine systems often do not integrate readily with the rest of a clinical organization's information infrastructure. Also, the nature of many current telemedicine system designs makes it difficult to extend them with additional clinical capabilities. They do what they do, and will only do more if there is a significant investment of time and money poured into engineering redesign. This means that system users may be locked into single vendor solutions and are not free to compose best of breed solutions.

Beyond these things, those vendors who make their living integrating these systems cannot readily adapt to the introduction of new technologies because each new technology may require a significant engineering investment to make it work in the vendor's existing systems. Suppose a vendor has already integrated an electronic stethoscope into his system. If a new stethoscope comes along that has better features, better performance characteristics, better pricing, etc. the vendor may not be able to readily offer this device as part of his system because the way in which the device is controlled and monitored is significantly different from the model that his currently uses and integration would require costly reengineering of this part of his system. In an ideal world, the vendor would be able simply plug the device into his system and those features that he has already addressed in his system would work just the previous model without

any reengineering. The only new work to be done would be aimed at addressing any new features that the device introduced to the system.

Finally, because of the way most systems are built today, they are expensive. As a consequence, organizations considering the use of telemedicine and those organizations who would pay for this service must consider whether they can realistically recoup their investments. In some cases the answer is a resounding "yes"; in others, it is much less clear. If there was some way for the integrator to save on material and engineering costs, a stronger case for ROI could be made and there would be a better chance of penetrating larger markets.

## 1.1 Why This Paper

While a number of factors contribute to the state of affairs described above, the issue of interoperability stands out as one that has the potential to positively impact many of the industry's current technical needs in a significant way. Even so, the question of working towards agreement regarding standard ways of building telemedicine system components is not without controversy. In an environment where some companies are struggling to make the next payroll, the possibility of "standards" being developed can be threatening. "What if the standards that are produced make my current systems 'non-compliant'?" "What if compliance with the standards requires an investment that I can't afford to make?" "Will the standards threaten my ability to differentiate myself in the marketplace?" "Will they limit my ability to innovate or to adopt emerging technologies that customers are asking that I address?"

Given these things, this paper is meant to be a vehicle for debate in the telemedicine community regarding the development of industry-accepted interoperability specifications. *Because the establishment of these sorts of standards cannot succeed without broad-based support from both vendors and users, this document is meant to be a strawman – a starting point for discussion and not a final answer.* It is expected that over time, as the ideas contained in this paper are debated and the industry begins to reach consensus, this paper will evolve into the architectural overview document for the set of telemedicine system interoperability specification documents embraced by the industry.

## 1.2 Organization of This Paper

The balance of this paper consists of three sections. The first section is a collection of utilization scenarios and system diagrams intended to describe, from a user's perspective, the kinds of capabilities that a telemedicine system based on interoperable elements might deliver and the kinds of components it might contain. This section also presents a set of system features that telemedicine system interoperability specifications would need to be able to address. The second section proposes a "straw man" telemedicine interoperability architecture. This section describes the architecture's logical composition and organization, the system's overall operation and the operation of its constituent elements, and addresses the question of how each of these elements might actually be implemented (e.g., what building blocks are available, how legacy features can be used). The final section of this document proposes a path forward – a set of

activities for the telemedicine community to pursue in exploring the issues related to telemedicine system interoperability and in moving towards consensus.

# 2 System Concepts

Telemedicine is about reengineering care delivery. The goal of this section of the document is to present a number of utilization scenarios and system configurations as a backdrop for considering how future telemedicine-based care delivery might operate in an environment where interoperability is a central feature of the systems.

## *2.1 Utilization Scenarios*

### 2.1.1 Tele-Home Care

Manny Garcia is a 42-year-old accountant who has just been diagnosed with diabetes. As his family has a history of heart problems and he himself has had increasingly high cholesterol levels, his doctor has told Manny that he needs to begin taking his health more seriously. The doctor has recommended that Manny try to lose about 50 pounds and that he work with his wife on changing their family's diet. As part of this process, Manny's doctor has hooked him up with the Provident Universal Management Plan (PUMP), a leader in the field of chronic disease management. In discussions with his "health advocate" (the title that PUMP gives to its case managers), Manny is given a list of equipment to procure that will he use with PUMP in managing his condition. These items can be procured directly through Provident or at a number of businesses in his area.

The next day, Manny buys his net-ready glucometer and weight scale from the local pharmacy, a combination pedometer / heart rate monitor from a sports equipment store, and a blood pressure monitor and electronic home blood test kit from a medical equipment supply company. Back at home, Manny successively pulls each of the devices from its box and pushes their start buttons. For each, a status indicator on the devices blinks while the device "auto-configures". A few seconds later, Manny's TV turns on and displays a setup program that has been retrieved from the Internet. By asking Manny a few simple questions, the program walks Manny through the process of integrating the device into Manny's home environment. Because Manny's home is equipped with a home gateway, home network, net-aware TV and videoconferencing camera, and computing resources, the only pieces Manny needed to create his new telemedicine home care system were the prescribed devices and the software that was automatically downloaded to run them.

From the start, Manny is able to interact on-line with his caregivers. When he contacts PUMP for the first time since setting up his system, his health advocate's caregiver station spends a little bit of time exploring the capabilities of Manny's "station". Since PUMP frequently deals with devices from a range of vendors, its system is already configured to handle all of the brands that Manny has procured. In contrast to this, when Manny has his first tele-visit with the nurse at his doctor's office, her system recognizes that it does not know a couple of his devices and downloads from the vendor servers the software needed to interact with Manny's system.

### 2.1.2 Rural Clinic

The Fairfield Clinic has several treatment areas that are outfitted with telemedicine stations. As a small town clinic, it cannot afford a regular staff of full-time doctors;

therefore, it has established a relationship with a large urban hospital a couple hours drive away. The hospital has its own network of telemedicine-ready caregivers stations dispersed across its various units and specialists in each unit rotate through on-call status for servicing these stations. In addition, it is not uncommon for individual caregivers to carry their own access devices (palmtops and the like) that they can use to access the telemedicine services on the hospital's network. During off hours at the rural clinic, a nurse practitioner triages cases that walk in the door, deciding whether or not to call in the local staff doctor or to draw on the specialists at the urban facility. When the latter is done, the nurse who manages incoming telemedicine calls for the hospital identifies the needed specialists and pages them automatically from his telemedicine terminal. As a specialist logs into his station or personal access devices, data from medical instruments, images from scopes and cameras, and audio from the on-going conversations is automatically piped to him. In cases where the access devices cannot handle a specific medium (e.g., full frame rate, full resolution video), the devices are able to negotiate for formats that they can handle. As needed, the patient's records, which are stored at the clinic, can be searched and reviewed by hospital staff. Once on-line, the specialists who have been assembled can conference with each other and with the nurse practitioner and, if needed, can request that an air ambulance be dispatched. If this is done, some of the instruments (e.g., infusion pumps, vital signs monitor) will travel with the patient from the clinic. All the while, data, imagery, and audio from these units (as well as units that may be added in the ambulance) will all continue to be forwarded to the hospital. Because the hospital supports a number of rural facilities in this way, its infrastructure allows for multiple simultaneous conferencing sessions. When all is said and done, all records generated by the encounter will be available to both the hospital staff and to the staff of the clinic.

### 2.1.3 Traveling Nurse

Jane Cantrell, a community health nurse, carries what she calls her "modern black bag" – a laptop and whatever medical instruments will be required for her day's work. Based on a knowledge of the kinds of patients that she will visit during the course of the day and on pre-visit phone calls to find out if there is anything in particular that each of her day's patients is concerned about, the nurse assembles the suite of instruments most likely to address the situations she will encounter. Because she is heading to places where she may or may not have good communications connectivity, Jane downloads to her laptop the records for each of her patients as well as identifier information for other patients that she has seen in the areas where she is heading today (from experience, her group has learned that you often see more patients than you had planned on these visits, so the identifier information allows these caregivers to more quickly record unplanned encounters). During her visits, Jane attaches and removes devices as required. The laptop automatically recognizes their presence and presents her with the user interface components needed to control these devices. As they are used, data from the devices is automatically logged into the appropriate records. On occasion, an alert generated by software agents vested with clinical expertise will appear on her screen offering an observation that they have made about data mined out of the current patient's record. As much of her job involves educating the patients for whom she cares, her laptop is also loaded with a range of video clips that address both the diseases of her clientele and the

care of these diseases. Not infrequently she finds herself hooking the video port of her laptop up to a patient's TV set so that they can watch a clip or two. Using a portable printer that she carries in her "black bag", she is also able to print out handouts that summarize for the patient what happened during the visit and what she is expecting them to do for themselves until she drops by again. As she ends her day back at her office, she connects her laptop to the office network. All records for the day are automatically uploaded to the office's central patient record server. Where applicable, a notice of key findings is sent to a patient's other caregivers (primary doctor, specialist, etc.).

### 2.1.4 Optimal Wellness

For Amy Tran, "excellence" is the watchword. In everything that she does, Amy gives 110%, and this includes watching her health. Even with the stress of starting her own business, Amy works to make sure that she is eating right, exercising, and getting adequate sleep and down time. Fortunately, Amy has found some help with this task in the form of a "personal trainer" system that she bought at the local sports store. In addition to an "intelligent" software package that helps her develop a whole life wellness plan and to execute on this plan, the system contains a number of physiological monitors aimed at the serious athlete along with a variety of devices that plug into Amy's home network in order to monitor her health habits. Under Amy's supervision, the system takes care of her menu planning and most of her grocery shopping (with groceries being delivered at a time that the system picks from her on-line schedule information and that Amy then approves). In addition to this, the system helps Amy think through the time commitments involved in any task or appointment that she is considering accepting and advises her when it appears that she is pushing herself too hard. As part of her exercise regime, Amy is training for an "iron woman" competition. Besides the training schedule that the system has helped her develop, Amy has access, through the system, to on-line reference materials and to sports medicine professionals who can advise her on a variety of topics from training techniques to care of injuries. On several occasions, Amy has found herself in a videoconference with one of these trainers and her primary care doctor. Recommendations from these sessions typically find their way back into Amy's system where they are used to modify training routines, sleep schedules, etc.

### 2.1.5 Visiting the Doctor from Home

Jim Ball is going to visit the doctor tomorrow, but for the first time in his life, he will not see the doctor in person. Jim is going to visit a doctor from his own home. In preparation for the visit, the doctor has asked Jim to log into the doctor's web site via a secure link the day before the visit. Once at the site, Jim enters some identifying information and is taken a set of screens that ask him about his health history. Where Jim is unclear on the question being asked, the system is able to provide more detail in the form of text, graphics, audio, and video clips. Following the history, the site begins to ask Jim about his current complaint. As Jim maintains his patient records at an on-line records company, this part of the interview is shorter than it might have been because the doctor's web site discovered the records and asked Jim's permission to extract history data from them. In the next part of the interview, Jim indicates that he is concerned about blotches that have appeared. Using more visuals aids, the web site leads Jim through a series of questions that help him identify the nature of these blotches ("Where are they

located?" "Do they look like any of these pictures?" "How long have you had them?" etc.). The site then notes that it sees that Jim has a net-ready digital camera and asks Jim if he would mind taking some snap shots of the blotches. As the site leads Jim through a series of shots, Jim's wife takes each picture for him. The web site then uploads each of the pictures into its notes and runs an analysis program to assess the adequacy of the shots. At the end of this process, Jim exits the site and is ready for the visit with the doctor the next day.

### 2.1.6 More Time to Live

Since childhood, Kathy Spellman has had a propensity to grow moles. When one turned up cancerous a few years ago, Kathy underwent a prolonged treatment regime. Since then she has regularly used a special camera-like device that her doctor recommended. Using multi-spectral imaging and highly sensitive range-finding techniques, the camera allows Kathy and her husband to regularly create a map of Kathy's skin. This handheld device, which contains a compact inertial measurement system, is "flown" over Kathy's body. The data stream from the camera is fed into a specialized processing and display unit that highlights a 3D human form on its display to indicate what parts of Kathy's skin have been surveyed. This guides Kathy's husband, allowing him to achieve full coverage. The processing unit, which maintains a map of the moles on Kathy's body, notes where new moles have appeared since the last recording session and where moles have changed since the last mapping session. At times, the system may highlight spots, using the 3D form on its display, where it wants a close-up shot. At these times, Kathy's husband brings the camera in close to the specified location and moves it around the area, providing images from different positions and camera angles. When a session is complete, rather than transmitting images of Kathy's entire body, the system transmits an updated list of all of the moles detected along with data that characterizes each of the moles in the lists (e.g., size, geometry, coloration, texture). As needed, Kathy's dermatologist will call her in to look at moles that he questions. While this process is sometimes obtrusive, Kathy and her husband both agree that it is a small price to pay for more time to live.

### 2.1.7 Full Circle

When "scientific medicine", with its hospitals and countless specialties and subspecialties, emerged in the early half of the 1900's, many caregivers lamented the increasingly impersonal nature of interactions with patients. As managed care horror stories were highlighted in the national news through the '80's and '90's and doctors and patients, both unhappy with healthcare's state of affairs, began to vocally demand their "rights", some in the healthcare community felt that it was time for a change. The medical community, they reasoned, needed to rethink the practice of medicine.

One such person was Jim Lansing, the CEO of Mid-Mountain, the largest care delivery provider in his region. In off-site brainstorming sessions with his key people, Larry homed in on the idea of trying to significantly restructure his care delivery processes by creating a new approach that was closer to chronic disease management in its structure than traditional acute care. The centerpiece of this new system would be a new kind of partnership between the patient and their primary care provider and between primary care

staff and the hospital. Based on a telemedicine-like communications foundation, these new partnerships would not simply erect a digital edifice that mimicked existing care structures; rather, the whole "floor plan" would be reevaluated. Working for months with his own staff and supporting researchers, Larry constantly asked why certain functions in the organization were done where they done. Did a given function have to be done by this group or this person? Who else could do it? Could any of the functions of the primary care office be moved to the home? How about the various specialist functions – could any of these be moved to the primary setting? What benefits would be realized by making these adjustments? What risks might accompany each change? Throughout, his goals were to help patients better care for themselves, to forge stronger partnerships between patients and their doctors, and to do what he could to free the primary care staff to focus primarily on delivering care.

When his plan was finally set, he took it on the road. After numerous meetings in which he shared his vision with policy makers, insurers, medical instrument manufacturers, and even some of the big information technology companies, he had money in hand to try out his ideas, a team of clinical experts and engineers to help implement them, and number of primary care providers and patients and specialists who would see if they could make his ideas work.

Building on existing home networking and computing technologies, Larry added a broad suite of diagnostic devices to patients' home to create sophisticated "medical portals" that allowed Mid-Mountain caregivers to do much more with "telemedicine" than just chronic disease management. With the benefit of an expanded suite of diagnostic and therapeutic instrumentation, decision support tools, "intelligent references", and a new hospital-based support organization to back up the staff in the primary care offices, the primary care staffs found that they were able to see more patients than before and felt that they were serving their patients better. Many of these caregivers naturally gravitated toward the role of "health coach". Because the system provided the primary care staff with access to Mid-Mountain's entire population of specialists and a mechanism for discovering who was available for consultation at any given moment, Larry noticed the bonds between the doctor's growing stronger. They soon prided themselves on their newfound ability to rapidly bring multi-discipline assessment and planning capabilities to the "hard" cases. Of course, none of this would have been possible without the seamless infrastructure that was put in place for person-to-person communications and information sharing.

Since the program's start it has been written up several times in the local press. The word in the community is that Mid-Mountain really cares about its patients and, as a result, companies are starting to switch their plans to Mid-Mountain and are asking how they can access Mid-Mountain's new capabilities. The word among the caregivers is that Mid-Mountain is a good place to work – "You have more freedom to do what you believe is in the best interests of you patient." Some of the doctors and nurses have started calling their system the "Mayo of the mountains". The nation is starting to notice as well. Larry has hosted numerous management teams from other hospital systems. Policy makers from the state and national levels are actively assessing Mid-Mountains approach.

Insurers are happy as well – Mid-Mountain's "whole life" approach is keeping their clients healthier and that means less money for them to pay out.

## 2.2 The Difference

So how do the pictures that each of these scenarios paint differ from what we know today in telemedicine? In the first scenario, one key difference is the plug-and-play nature of the devices that Manny bought. Whereas in Manny's case the devices were automatically configured into the house once they were discovered, today we typically use programmers to stitch these devices into our systems. Next, Manny wasn't told to go buy a specific device model from a specific company. He chose his devices based on whatever criteria were important to him (cost, size, appearance, reputation, etc.) and this was okay. Today, systems are often constructed with specific device brands and models in mind and plugging in devices from another manufacturer at will is unheard of. Third, some of the capabilities (e.g., blood analyzer) that Manny has in his home are found today only in clinical settings. Others (e.g., pedometer), which will be used by professional caregivers supporting Manny, are not normally considered to be part of the clinician's toolkit. A fourth difference is the use of resources (e.g., TV and home network) that are already in the home for other purposes to support telemedicine. A final difference is the ability of nodes in a telemedicine system that have never before seen each other to explore each other's capabilities and to then seamlessly interoperate.

The rural clinic scenarios exists to talk to several system features of value to future telemedicine systems. The first is the ability to establish multiple "nodes" within a given care delivery location. In this case, the clinic's telemedicine stations may be nothing more than user interface and medical device capabilities along with some way of associating the various resources at a station with one another. In all likelihood, in a situation like this, patient record storage would be centralized on a server somewhere in the clinic, as would the clinic's wide-area communications and, potentially, its source of clinical applications and reference materials. In this scenario, there is a need to dynamically conference in users who may be using a range of devices to access the conversation (e.g., one is a workstation, another is using a PDA, while a third is using only a phone). The movement of instruments with the patient from clinic to ambulance to hospital is meant to talk to the need to think of telemedicine "stations" as a collection of resources that federate for a time to support some clinical task and that can change their makeup (in this case, which communications they are using to talk to the hospital systems) on the fly. The use of a shared communications infrastructure in the hospital as well as the ability to simultaneously handle multiple remote clients speaks to the need for security. The "applications" are resource-aware, knowing when the resources that they have acquired can support the task at hand.

The traveling nurse scenario is meant to say that the platform is just a vehicle and that it is the devices that give it personality or purpose. This scenario also points out that even records are a "service" that can be accessed both locally (on the PC) and by other devices (like the record server in the office). Telemedicine systems can also deliver clinical intelligence.

In the case of Amy Tran, the first point is that future "telemedicine" won't just be for acute and chronic care. Next, it is meant to say that, in the future, intelligent software can act as both the patient's and caregiver's agent. Third, tomorrow's systems will be able to integrate with other, non-medical kinds of systems (e.g., ordering groceries). In addition, in this environment, much of the education is geared toward the patient. Finally, when Amy's doctor looks at her records, there will be elements (such as her fitness level figures) that are not typical for current medical records.

In the next scenario, the point is simply that some of the things that we assume that the professional must do may be transferable to the patient if we provide the right support. The Kathy Spellman scenario says the same thing. It also points out that our future instruments might be quite intelligent.

In the final scenario, the point is that tomorrow's telemedicine is about more than simply copying existing care structures onto a technological foundation; rather, it is about thinking through how we would like to deliver care and then determining if technology can support it or not.

## 2.3 Features We Want Our Stations to Incorporate

Figure 1 is the functional block diagram for a typical telemedicine-based home care system. The system's patient station provides ***user interface*** components (camera, display, microphone, speakers, controls, and status indicators) to support human-machine interactions and person-to-person communications. It also provides ***instrumentation*** (e.g., physiological sensors) that enables the caregiver to monitor the patient and,

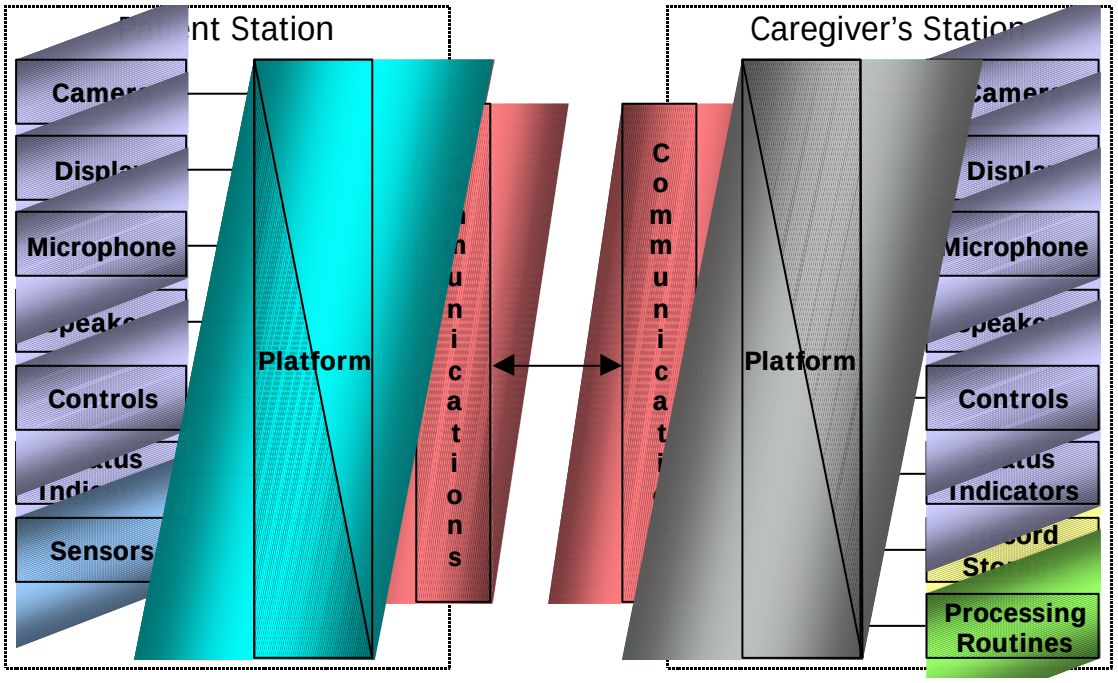


**Figure 1. Home Care System Block Diagram**

possibly, the patient's environment (e.g., for temperature, allergen levels, etc.). ***Communications*** components exist to allow the patient station to transfer data to and from the caregiver's station. Everything is tied together in the station by a computing ***platform*** that provides both low-level services (e.g., moving data between the various other resources in the station) and high-level services (e.g., control of the station operation). On the caregiver station, the organization is largely the same; however, it does not contain any sensors and adds two additional functions: ***record storage*** and ***processing routines***. The first exists to allow the caregiver to maintain on-going records of a patient's state of health. The second consists of those routines used by the station to digest or transform the data that it collects (e.g., one such routine might be used to determine if a given parameter is trending in a certain direction).

While the example shown is for a home care system, the services provided by the system (user interface, instruments, communications, low- and high-level platform, record storage, and processing) are common to many telemedicine-based care delivery systems. The balance of this chapter addresses each of these services in more detail, looking specifically at what we would like to be true of the components that make up each service in order to deliver the kinds of operational capabilities described at the start of this chapter.

### 2.3.1 User Interface

The user interface portion of the telemedicine system exists to allow a person to:

- receive clinical and other kinds of data,
- supply data to applications running on the system,
- control various aspects of the system's operation, and
- monitor the operational status of the system and its constituent elements.

In building our next generation of telemedicine systems, there are several features that we would like the interfaces to possess. In particular:

- it should be easy to replace interface components as the clinical needs or user preference dictate
- our designs should allow for the user of multiple simultaneous interface devices within the context of a single telemedicine "station"
- the interface functions of the system should be readily extensible

#### *2.3.1.1 Easy to Replace*

In most systems today, the user interface is tightly bound to the system. While a system may offer multiple interfaces (e.g., a standard GUI and a web interface), the systems generally lack the ability to swap out interface elements at will. Figure 2 is a notional view of what we would like to be true instead. In this figure, the "rest of the telemedicine system" is able to detect the presence of a new interface device, to explore its capabilities, and to then deliver content in a format appropriate to the device. Ideally, this process should occur as close to the time of use as possible. Doing this, when the station containing the devices boots might be okay but "hot swapping" of devices would be preferable. It should be noted that this ability to detect a "new interface device" applies not only to new kinds of devices but also new occurrences of the same device

(e.g., as when one doctor had been using a PDA with the system leaves and then a new doctor with the same kind of PDA appears).

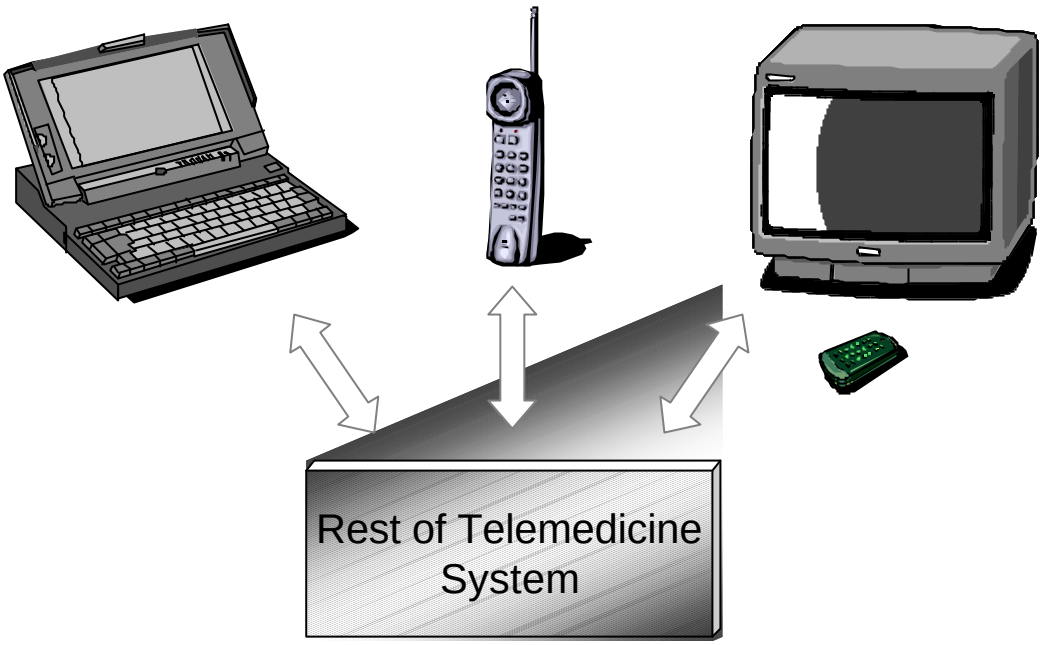


**Figure 2. Easy to Replace Interface Components**

### *2.3.1.2 Multiple Simultaneous Devices*

In certain situations, it will make sense for a system to interact with a user through multiple interface devices used all at once. Figure 3 shows this sort of situation. In this example, a hospital bed is equipped with a wall-mounted speaker and high resolution

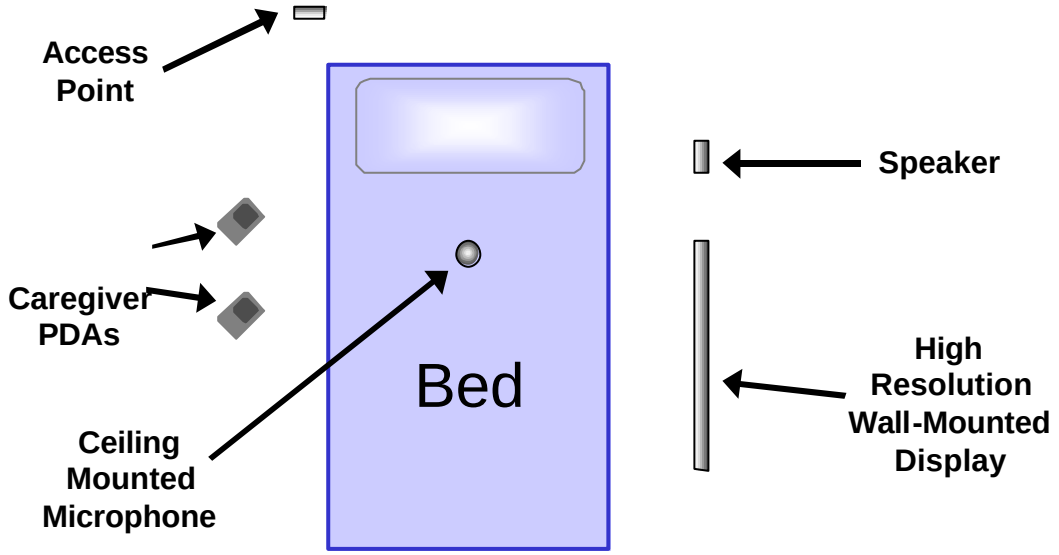


**Figure 3. A Station with Multiple Interfaces**

display. A microphone over the bed offers one means for both patient and caregiver to issue commands to the system. An access point on a wall next to the bed allows portable devices carried into the room to connect with the rest of the resources in the patient station. One utilization scenario for this system might involve a doctor and resident entering the room carrying their PDAs. The patient is watching a television show on the display and the listening to its audio on the speaker. As the doctors enter the room, the access point recognizes the presence of the PDAs and automatically uploads the patient's "summary sheet" to the PDAs. This sheet serves as a digest of the patient's recent case history and an index into the patient's entire medical record. Pointing to an item on this sheet, the doctor says out loud, "Computer, show series on the wall display." The command, which is picked up by the microphone, is combined with the current selection on the PDA and immediately an x-ray series of the patient's chest is displayed along side the TV display. "Computer, enlarge number 6 and hide TV". Image number 6 now fills the screen. After some discussion, the doctor retrieves an electronic stethoscope that, as with the PDAs, is recognized by the access point. With a few taps on his PDA's screen, the audio output from the stethoscope is routed to the speaker. "Computer, analyze stethoscope output and show results on the wall display." After a few moments, a computer-looking form appears on the display with the findings of the analysis.

#### *2.3.1.3 Readily Extensible*

As a final objective, we would like for the user interfaces to be dynamically extensible. When the need arises to view a new kind of data or to monitor and control a new feature of the system, we should be able to install the necessary interface features on the fly. For example, in the case of Manny Garcia, the addition of a new device (e.g., a glucometer) means that the interface in Manny's home (the TV) would display controls that Manny can use to both operate the device and read the data that it generates.

### 2.3.2 Instruments

The instruments in a telemedicine suite exist to allow the suite's users to acquire clinical data or to deliver therapeutic treatment. While these instruments have traditionally been medical instruments, it is very likely that future care delivery systems may extend this model to include other non-traditional devices that somehow relate to a person's health. Examples of these sorts of devices might include those that provide insight into a person's sleep habits, dietary habits, or living environment (normal temperatures, noise levels, air quality, etc.). Depending on the nature of the device, a user may need to be able to configure the device, control its operation, and/or monitor its status. In addition, in this future world:

- it must be easy to add and remove devices from a station,
- a station must be able to handle multiple simultaneous devices, and
- a station must be able to handle devices in a vendor-independent fashion.

#### *2.3.2.1 Easy to Add and Remove Devices*

In current telemedicine systems, it is common to find dedicated connectors for the various devices that the systems host (e.g., "The pulse oximeter plugs in here; the blood pressure cuff goes right here; and this connector is for the weight scale."). Although common, this approach is quite limiting. If a customer wants a different suite of

instruments or additional instruments beyond those already supplied by the telemedicine system, the new capabilities will not be added without a significant amount of reengineering of the system. What would be preferable is some approach that allows for the dynamic addition and removal of devices as needed to suit the system user's own unique situation.

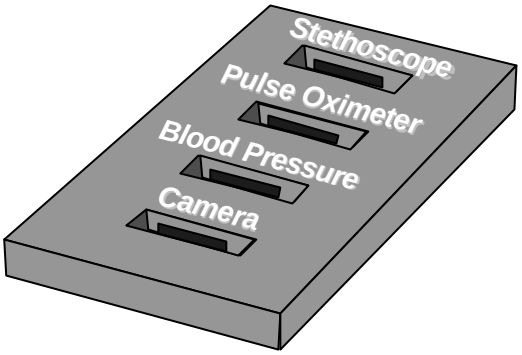


**Figure 4. Dedicated Connectors**

#### *2.3.2.2 Multiple Simultaneous Devices*

In certain situations, there will be a need to add multiple devices of a given kind to a system. For each, during infusion therapy, there may be multiple infusion pumps at the bedside. The telemedicine stations that we build should allow for this situation and should make it simple for a remote user to distinguish between multiple instances of a device when they exist a site that is being monitored and controlled.

#### *2.3.2.3 Vendor Independence*

While individual devices may vary in the specific suite of functions that they deliver to a system, like devices ought to be able to deliver a core common set of data and respond to a common set of commands. For example, if blood pressure cuffs from two different vendors are to be used with a telemedicine system, they should both be able to respond to "Start" and "Stop" commands and both be able to return systolic and diastolic pressures. Even if they each have their own unique features (e.g., the ability to queue up many readings or the ability to adjust the maximum inflation pressure of the cuff), at a minimum, their core behaviors should be identical so that they can be readily used in place of each other in a system.

### 2.3.3 Record Storage

In order for telemedicine to support a wide variety of disciplines and utilization concepts, we need to be able to impose several specific requirements on the mechanisms that we use in these systems to store and retrieve patient data. In particular,

- we must allow for a range of storage locations relative to their point of actual use,
- the records must accommodate a wide assortment of data types,
- the records must be extens ible in terms of the kinds of data that they can store,
- they should be media independent, and
- the record systems that we use should (ideally) be "multi-lingual".

### 2.3.3.1 A Range of Storage Locations

As shown in Figure 5, a telemedicine system can interact with record storage mechanisms located in a variety of places. For traveling nurse or self-care scenarios, local storage is a

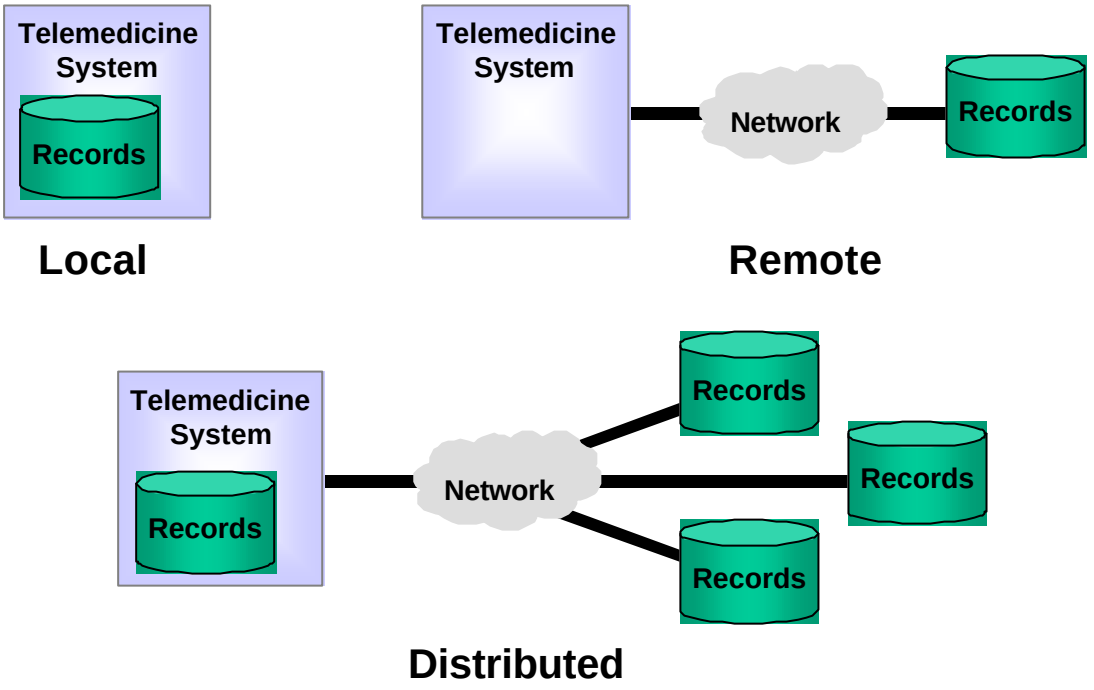


**Figure 5. Different Storage Topologies**

reasonable choice. In others, the repository used may be remotely located. In still others, the records for an individual may be scattered across multiple local and remote storage locations. Irrespective, from the point of view of the other services in a telemedicine station needing to access these records, the fact these different topologies are possible should not affect how these resources perform their own functions. Finally, in some situations, the question of which repositories are being used at any given moment will be quite dynamic. For example, in a doctor's office, the record that he sees for one patient may be drawn both from his office's own record server as well as from servers belonging to other organizations. For the next patient, all of the records may be stored in the office's server and for the patient after that all of the records may be retrieved from remote servers.

### 2.3.3.2 An Assortment of Data Types

In order to support a broad range of system capabilities, the record storage portion of a system needs the ability to store a range of record types including, among other things, text, numeric data, waveforms, still images, videos / cines, audio and voice, and solid models.

### 2.3.3.3 Extensible

Because one of our goals is to allow a system user to add new kinds of devices at will, it needs to be easy to extend a telemedicine station's patient record storage system to handle the kinds of data generated by these new devices. Ideally, this could be done dynamically, as the need presented itself. Once extended in this way, the record system should make it easy for other resources to discover its ability to handle a new data type and to both write and read data items of this type.

### 2.3.3.4 Media Independent

Patient record systems can be built on a number of different types of building blocks. These include the physical storage media (hard drive, smart card, etc.), the "file systems" used to control reading and writing from these media, and the databases and similar mechanisms used to allow sophisticated approaches to storage and retrieval of information. From the perspective of other services in a telemedicine station, the exact means used to deliver record storage should be transparent. At the same time, when certain key events occur (e.g., the storage device runs out of space or is removed from the system), other services dependent on the storage service should be able to learn of these events and to respond accordingly.

### 2.3.3.5 Multi-lingual

An old joke says that the nice thing about standards is that there are some many to choose from. Nowhere in the telemedicine world is this more evident than in the area of patient records and, for the foreseeable future, this is very likely to remain the case. Given this,

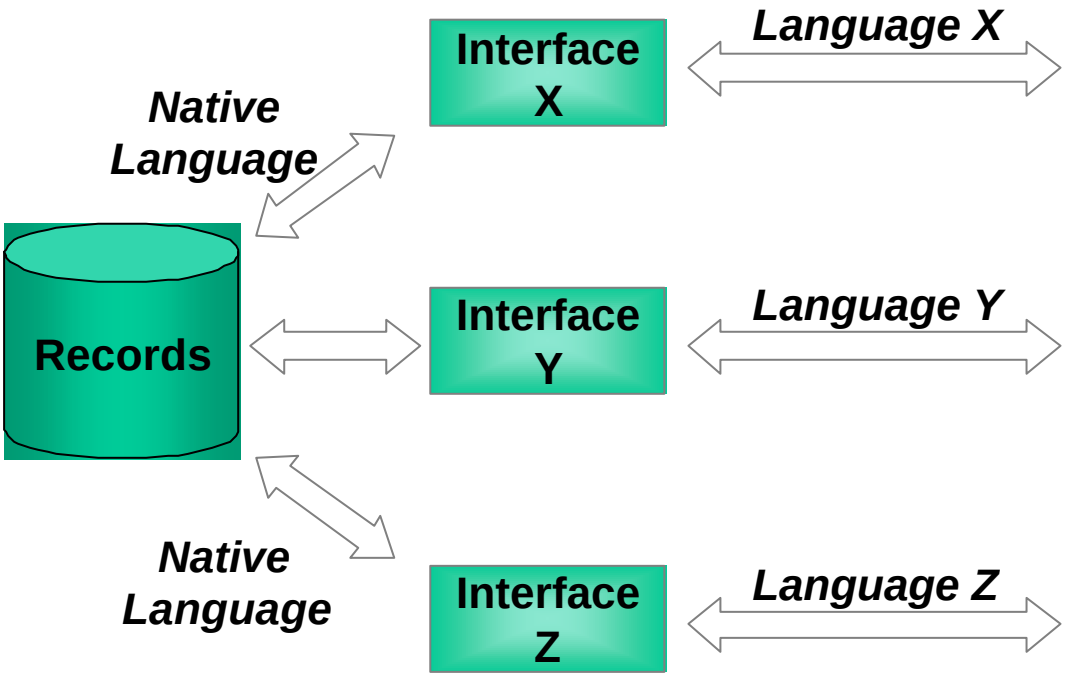


**Figure 6. Storage Language vs. Data Exchange Language**

it would be good if a system that we build had some means of detecting what medical data exchange languages and protocols are employed by any given system with which our system connects and to then be able to negotiate with the other system the specific exchange mechanisms that will be employed while our system and the other are sharing data with one another (Figure 6).

### 2.3.4 Processing

Whereas the service areas described so far all deal with the flow of "raw data" in and out of a telemedicine station, many of the interesting things that can be done with these systems depends on the ability of the system to be able to transform data. To this end, processing components exist to deliver both mechanistic data transformation capabilities (e.g., extracting the statistical profile of a set of data) and higher levels of intelligence (e.g., assessing the suitability of a data set for diagnosis) to a telemedicine system. While components belonging to this class may exist only as software (unlike the others described so far which have a physical dimension as well), we will want our processing components to share many of the same characteristics as components belonging to the other service areas. In particular, we want them to be:

- dynamically configurable
- distributable

#### *2.3.4.1 Dynamically Configurable*

In certain settings, there is value in being able to select at the time of use the processing approach to be employed for a given task. Given this, whatever approach is chosen for building telemedicine systems should allow for the ability both to add and remove processing components from a system and to select which component of a given type will be used for the task at hand (Figure 7).

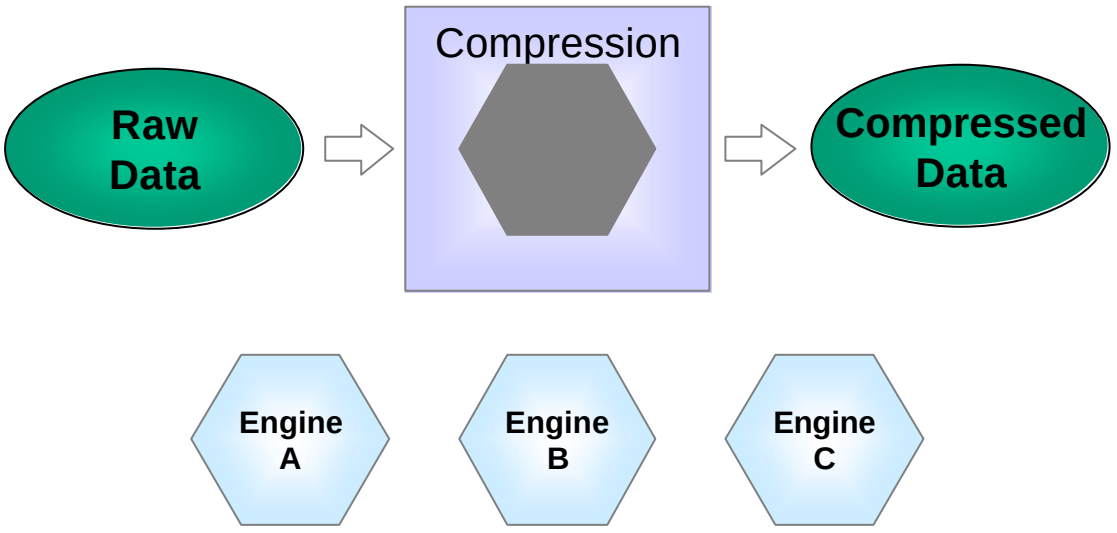


**Figure 7. Interchangeable Processing Components**

### 2.3.4.2 Distributable

As with record storage, it should be possible for a telemedicine station to draw on processing resources that are local to the station and those that are remotely located. In Figure 8, the Process M draws on Process N in order to do its job. From a functional perspective, Process M should not care if Process N is local, is distributed out over some part of the same local area network at itself, or is hosted by a third party service provider.

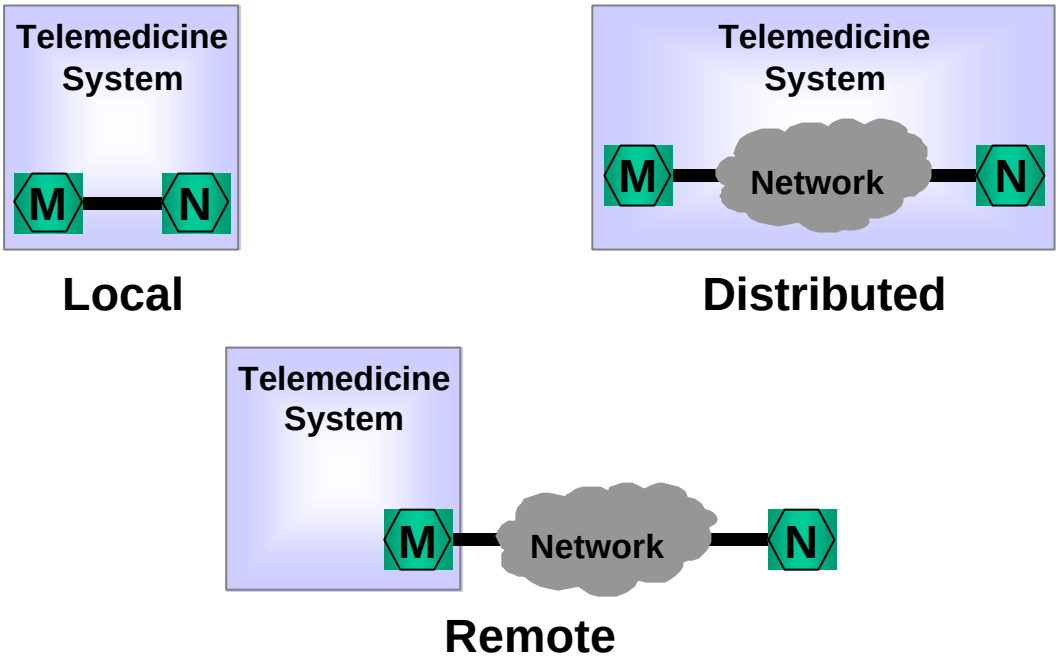


**Figure 8. Distribution of Processing**

## 2.3.5 Communications

If we call a collection of resources that function together as a whole in one location a telemedicine station, then the communications resources that are part of this collection exist to allow the station to interact with resources that reside outside of the station. In this role, communications resources support both data transfer and person-to-person communications. Attributes that we would like these resources to display include:

- the accommodation of a range of devices and formats
- the ability to shield the users of a station's bandwidth from the details of how to acquire and manage bandwidth from a device
- support for demand-driven allocation of bandwidth

### 2.3.5.1 A Range of Devices and Formats

In developing our telemedicine systems, one thing that we must not do is to tie ourselves to specific technologies that are likely to be replaced over time by newer technologies. In the world of communications this means that our designs should be able to support a range of devices (POTS, ISDN, Cable Modem, etc.) and a range of protocols that run on top of them for the purpose of moving data.

### 2.3.5.2 Shielding Users from the Device Details

Any given communication device will very likely employ its own mechanisms for controlling and monitoring its operation and, potentially, for inputting and outputting the data that it exists to transport. To the degree possible, those resources within a telemedicine station that use the station's communication capabilities should not be required to know the specifics of how these devices operate. As Figure 9 illustrates, the station's communication services should present these resources with a standard means of acquiring channels to the outside world but should hide from these resources exactly how this is done. Ideally, the Bandwidth Client has no idea what kinds of devices and how many exist beneath the Bandwidth Manager.

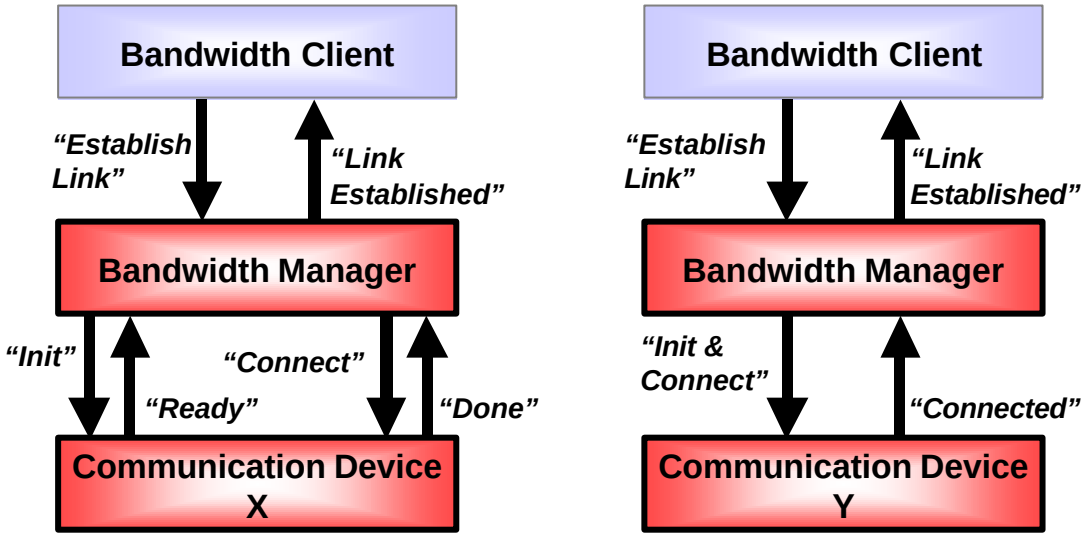


**Figure 9. Hiding the Details of How a Device Works**

### 2.3.5.3 Supporting Demand-driven Allocation

Because a given resource's need for bandwidth will vary as a function of the clinical operations that are underway at any given point in time, the communications resources in the system should be able to dynamically specify how much bandwidth is available to each bandwidth client at any given moment in time (Figure 10) and to allow clients to know when demand exceeds supply.

## 2.3.6 Platform

In future telemedicine systems, the "platforms" which serve as the foundation for these systems may be more virtual than physical. The platform exists in these systems to:

- provide connectivity between components belonging to the station's various service areas,
- to provide some means for these components to form an association, and
- to deliver or coordinate whatever services are required to ensure the secure operation of the station.

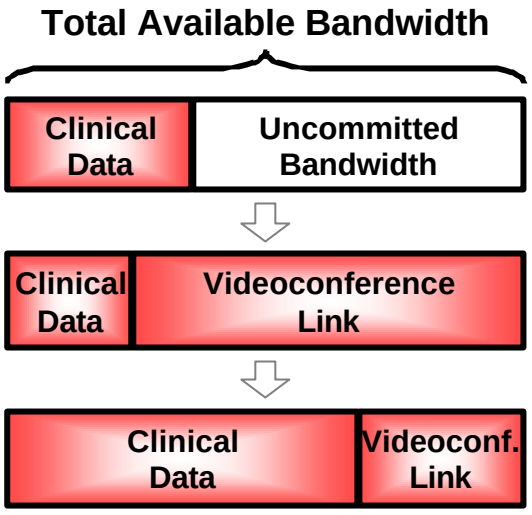


**Figure 10. Changing Allocations with Changing Needs**

As such, while a platform may be a single device (such as a personal computer), it may also consist of multiple devices (e.g., "dumb" and "intelligent" access points) and network segments. For example, in tomorrow's home, it will not be surprising to see a platform that consists of a gateway device that connects multiple networks within the home (e.g, the audio/video net, the computing net,…) with the outside world and that also serves as a computing platform for the services that run on the home network.

### *2.3.6.1 Connectivity*

The first function of the platform is to allow the various components that exist at the station location to communicate with one another. This differs from communications just discussed in that the focus here is on moving information between components that are collocated within a station rather than components that located away from the station. In networked designs, this connectivity may be achieved largely through the use of hardware elements. In single platform designs, this "backplane" may consist almost entirely of software. Depending on the nature of the components that are being connected and on the function being performed by these federated components, this

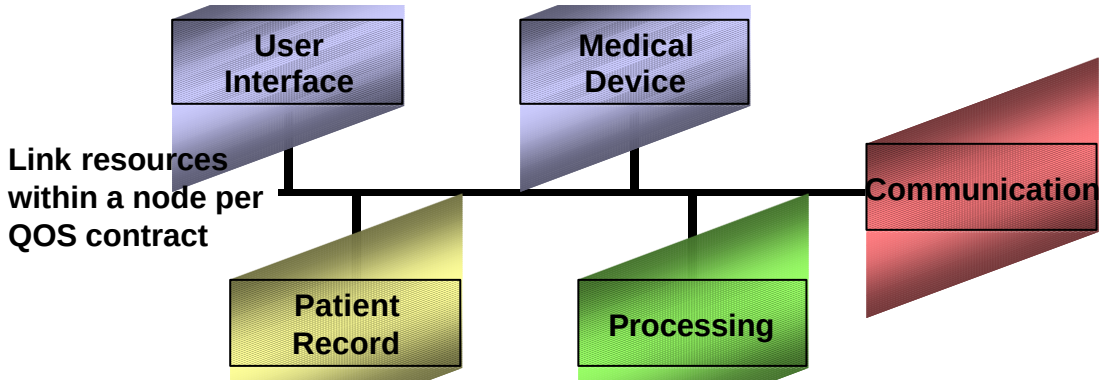


**Figure 11. Connecting the Components**

backplane may need to be able to deliver to the components certain quality of service guarantees (e.g., maximum latency, minimum bandwidth).

### 2.3.6.2 Association

As shown in Figure 12, it is possible for many telemedicine system components to exist on the same local communications infrastructure and yet not be part of the same "station". In the figure, each exam room is equipped with a set of medical devices and user interface mechanisms. In addition, the doctor's office maintains its own patient record server and application server (which hosts a number of clinical and administrative software packages as well as on-line reference materials). The doctor keeps a terminal in his private office that provides access to all of the data and applications maintained on these servers. Finally, all communications between resources on the office's local area network and resources outside of the office are mediated by the communications interface that serves as both a router and firewall of this network.

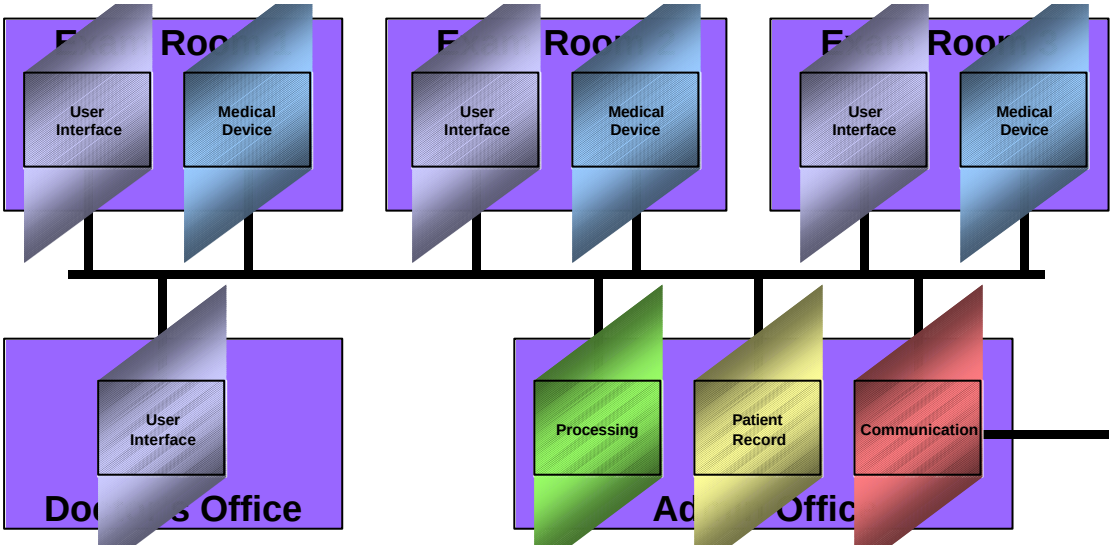


**Figure 12. Establishing Associations**

In the environment shown, a telemedicine station will very likely consist of all of the devices in a room (exam or doctor's office) along with other resources needed to perform the task at hand (e.g., a place to store data from the medical devices or processing routines needed to analyze this data). It may even include resources located outside of the doctor's office (e.g., this would be true if the records for a patient being seen were scattered across many servers or if the doctor subscribed to a service that maintained its clinical applications on its own computers rather than installing them on the applications on its client's machines. Given this situation, one of the functions that the platform or "backplane" must perform is to establish associations between system components so that, for example, the doctor, in trying to use a medical device from Exam Room 2, does not accidentally end up interacting with the same kinds of device contained in one of the other exam rooms.

### 2.3.6.3 Security

During the process of association of resources within a station, one of the key needs is that this be done in a secure fashion. Among other things, this will mean ensuring that the data flowing between station components cannot be viewed by unauthorized entities, that system functions are controllable only by entities allowed to invoke these functions, that even in storage the data cannot be stolen or lost, and that data is not corrupted as it moves from component to component within the system.

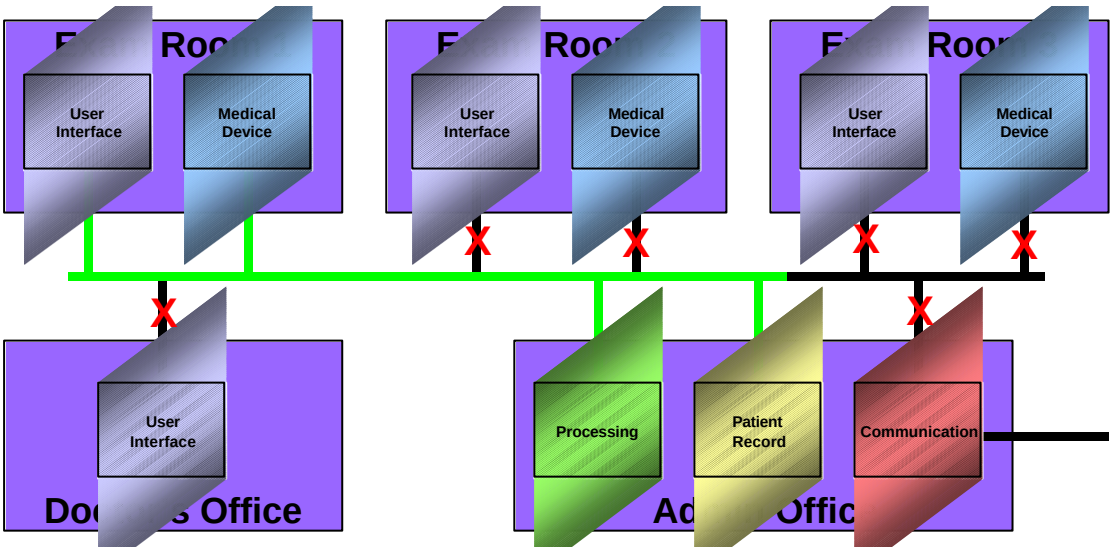


**Figure 13. Securing the System**

## 2.3.7 Protocols

Finally, a telemedicine station needs some means of allowing its users to control how various resources are marshaled in support of specific functions. Ideally, this would be done on a user-specific basis and in a way that supports intelligent use of system resources.

### 2.3.7.1 Control How Resources Are Marshaled

While the job of components belonging to most of the services described above is to generate or accept information flows, something else is needed in order to tell these components how to connect to each other for whatever clinical purpose is at hand. For any given operation that is to be performed, there will be a need to acquire the needed components, stitch them together, and then watch for key events (such as the removal of a resources from the system) that will require that intervention. Once an operation has run its course, the assembled components will need to be released back to the system for reuse.

### 2.3.7.2 Allow for User-Unique Preferences

Just as in computing, where users will often customize their computing environment, users of telemedicine stations may want to customize the ways in which they use their systems. For example, while both configurations in Figure 15 allow their users to acquire

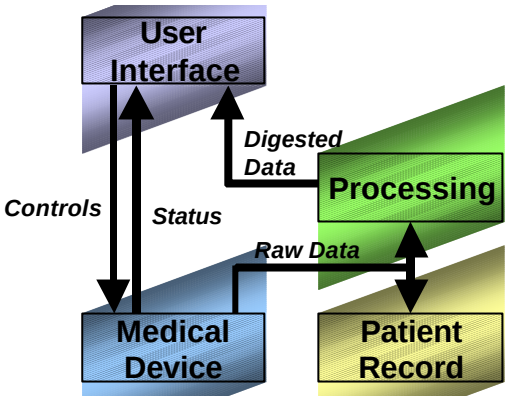


**Figure 14. Connecting the Components**

a given kind of medical data, one returns only the digested data whereas the second returns both raw and digested data (as reflected by the user interfaces). In addition, the one user prefers to use one process to digest data, while the other prefers to use a different process.

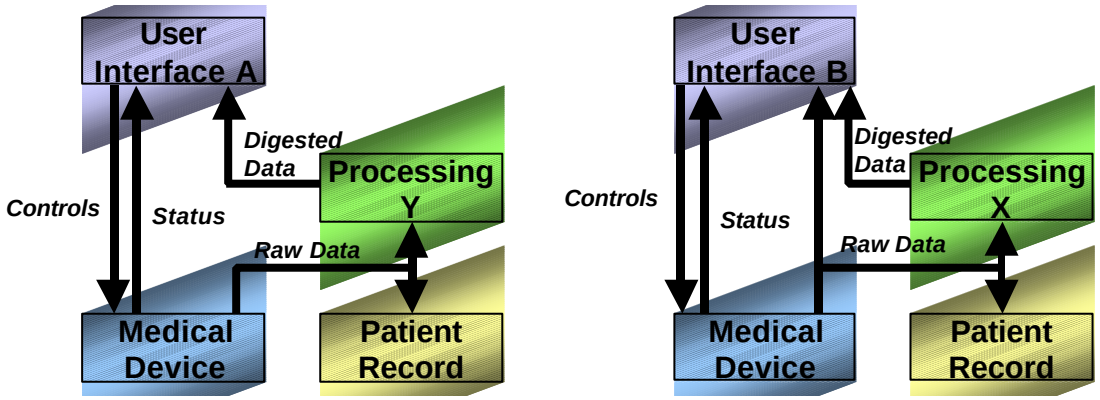


**Figure 15. Same Function, Different Implementations**

### *2.3.7.3 Intelligent Use of System Resources*

Because systems of the sort described so far are not composed of fixed combinations of resources, whenever a new operation is initiated, the system needs to determine whether the resources needed to support the operation will deliver the desired performance characteristics and, if not, whether there are acceptable fallback positions. For example in Figure 16, the medical device produces uncompressed data at a rate of 60 megabytes per second. While the user interface can handle the images produced, the communications cannot support this throughput. As an alternative the system may recommend buffering the data to the patient record (there is enough room for a little over 5 minutes) and then forwarding the data over the 10BaseT connection after the fact.

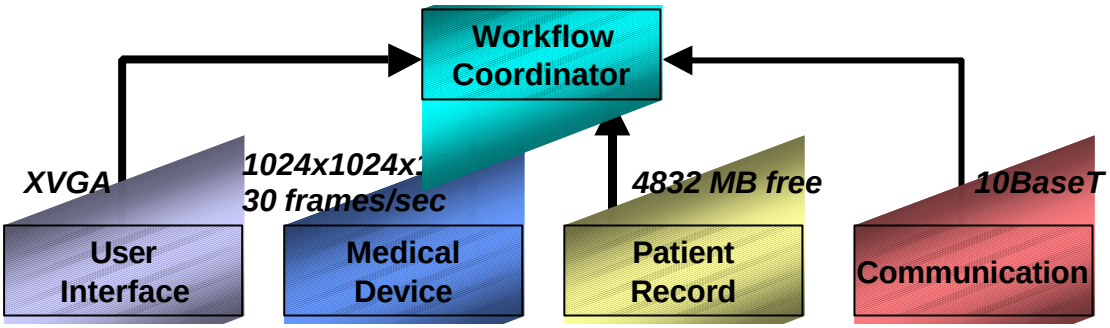


**Figure 16. Surveying Component Capabilities**

## 2.4 Features for the System to Incorporate

If Figure 1 typifies today's telemedicine systems, then Figure 17 depicts what is likely to be true of tomorrow's. Rather than a closed network in which all services are contained within the network's stations, this system consists of a suite of services that are distributed across a range of devices and locations. Just as devices within a station must be able to locate and federate with each other, stations within a network should be able to associate with each other to carry out clinical functions. To achieve this, the telemedicine systems need to support:

- resource location,
- discovery,
- resource acquisition,
- conferencing,
- quality of service, and
- security.

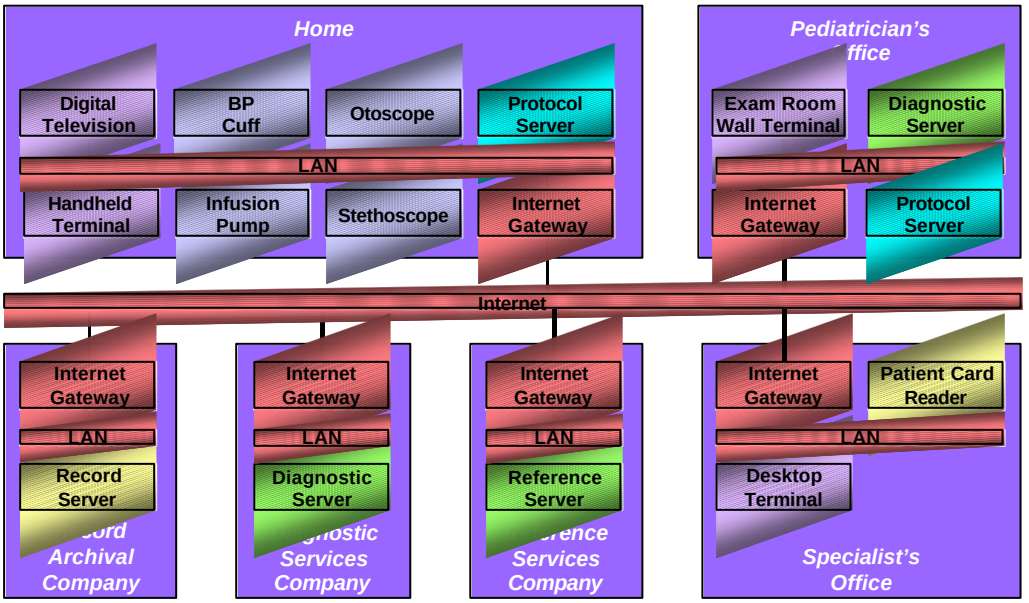


**Figure 17. A Distributed Telemedicine System**

### 2.4.1 Resource Location

Before stations can begin to interact, they first need to be able to find each other. In particular, infrastructure services must exist that allow a station to locate a specific station as well as to locate stations that provide certain kinds of services (Figure 18).

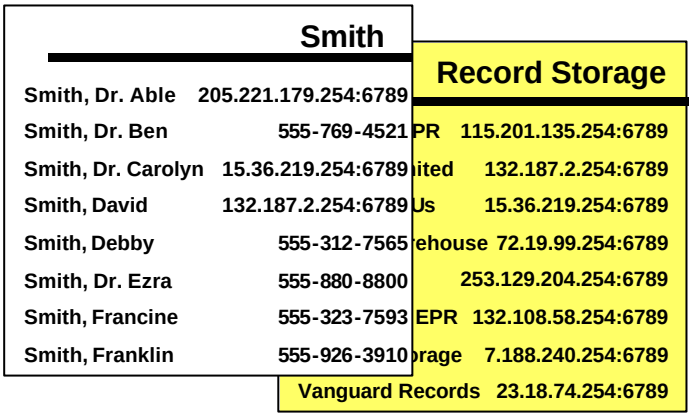


**Figure 18. Station Location Information**

### 2.4.2 Discovery

Because we are trying to create systems that allow independently designed stations to interact and because we want to allow these stations to be dynamically composed (i.e., end users can add and remove station capabilities), stations that locate one another need the ability to explore each other's capabilities (Figure 19). In addition, depending on the nature of the stations, each station also needs the ability to decide on a case-by-case basis which capabilities will be exposed and which will not.

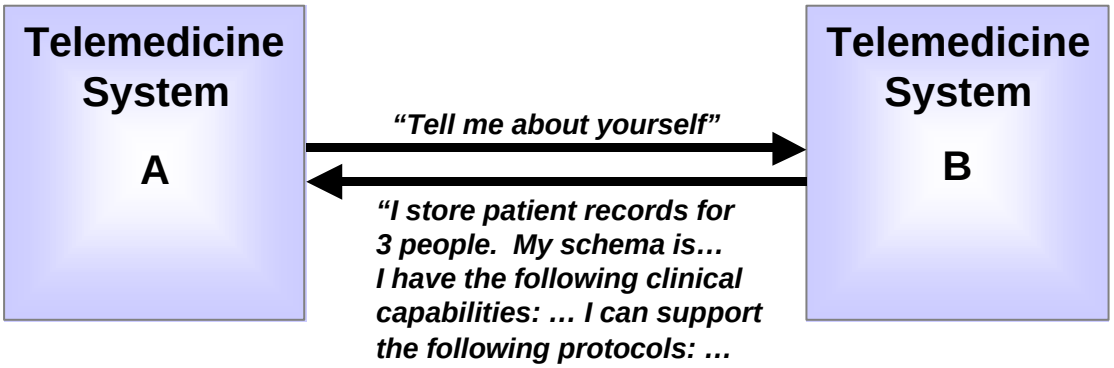


**Figure 19. Surveying Component Capabilities**

### 2.4.3 Resource Acquisition

Once a station knows what services another station offers, it may want to make use of one or more of these services. In order to avoid conflict with other stations that may want to access these services, the system must provide some means for allowing the requesting station to reserve another station's service for the duration of the time that it is required

and to then release this service for other stations to use. In addition, in order to allow dynamic configuration of stations, systems will need the ability to locate resources that support a given device or function and to make these available to the stations that need them (Figure 20).

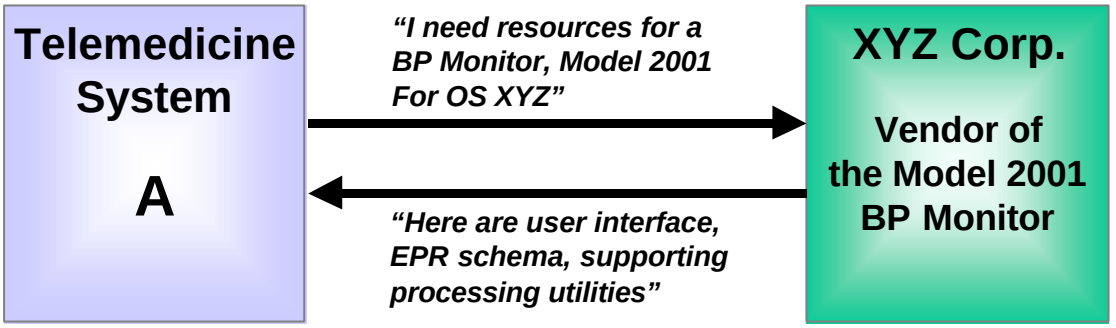


**Figure 20. Acquiring a Resource**

### 2.4.4 Conferencing

In constituting our systems we also want the ability to support multi-party interactions in which stations can be dynamically added to and removed from the conference. This should include not only the ability to support audio and video conferencing between individuals, but should also the formation of systems in which services are spread across a number of stations (e.g., patient records are served from one, clinical data is acquired from another, diagnostic routines on third, and everything is controlled from a fourth).

### 2.4.5 Quality Of Service

In certain applications, it is critical that the stations taking part in a transaction and the networks that support them be able to negotiate quality of service guarantees.

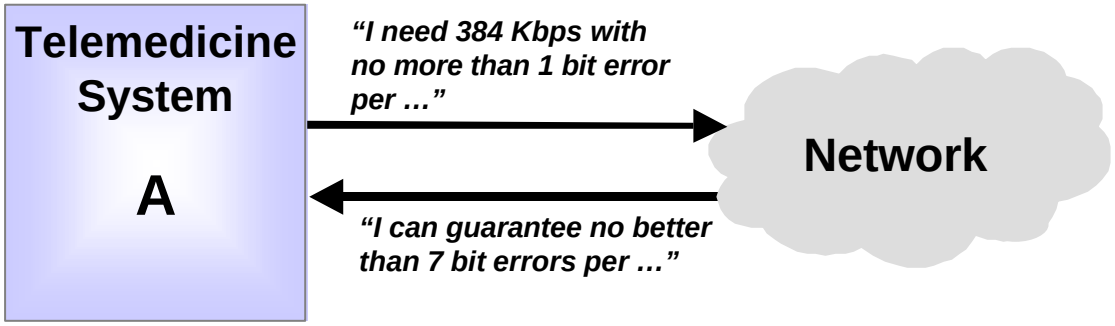


**Figure 21. Negotiating Quality of Service**

### 2.4.6 Security

Finally, the kind of system being described here must also be able to address a range of security concerns. Individual nodes must be able to ensure their own integrity and to determine and react when they have been compromised. A station trying to establish a

relationship with another station must have some means of determining whether that station is trustworthy.

While a number of other requirements could be enumerated here, the most important to note is the ability of stations to explore each others assurance capabilities and to negotiate a joint security policy each time an association is established (this is in contrast to much of what is happening today in the informatics world where organizations are establishing fixed, point solutions to these kinds of problems).

## *2.5 Other Considerations*

Before closing this chapter on what we want our systems to do for us, a few other issues need to be noted. In particular, we want:

- our systems to be compact, rugged, affordable, and easy to use,
- to support a variety of care delivery system operational concepts,
- our solutions to be as technology neutral as possible,
- to avoid reinventing the wheel, and
- to create solutions that we can grow into.

### 2.5.1 Compact, Rugged, and Affordable

While modern medicine has a wide range of capabilities that it can bring to bear on clinical problems, many of the mechanisms used to deliver these capabilities are geared for use in traditional settings. As a consequence, these devices may be large, fragile and expensive (as compared to consumer devices), and may require an expert to operate. If telemedicine is to realize the promise of "anyone, anywhere, anytime", then much thought will have to be given to how these capabilities can be repackaged in ways that allow them to be used outside the traditional clinical environment and that make these devices accessible to a larger population of users.

### 2.5.2 Variety of Care Delivery System Operational Concepts

While the goal of this work is to standardize the approach taken to inter- and intra-station interoperability, the specifications developed to support this should not (to the degree possible) constrain the kinds of care delivery approaches that might be implemented. In particular, the architecture should allow for:

- dynamic association of stations and patients
- dynamic association of stations and caregivers
- multiple patients per station and multiple stations per patient
- multiple caregivers per station and multiple stations per caregiver

### 2.5.3 Technology Neutral

To the extent possible, our solutions should not depend on the existence of specific hardware or software technologies. Ideally, replacement of specific elements of a system should be transparent to the rest of the system.

### 2.5.4 Avoid Reinventing the Wheel

In developing our architectural approach, we must do all that we can to make use of what already exists whenever possible and to invent new things only when required. To this

end, we should draw on the work of existing standards development and related organizations as a basis for the solutions that we propose to deliver.

### 2.5.5 Solutions We Can Grow Into

There are already a lot of useful components out there for building telemedicine systems; however, none will already support the full vision of what we want to achieve. Our solutions should not force users and developers to scrap everything that is already in place; rather, we should make it easy for these people to start where they are at and to grow their systems to a point where they are compliant with the long-term interoperability goals set by the telemedicine community.

# 3 An Interoperability Architecture

The goal of this chapter is to propose a strawman architecture that supports the capabilities described in the previous chapter and to identify options for implementing this architecture. As we in the telemedicine community are in the very early stages of discussing what interoperability might mean for us, it is not expected that this proposal will be embraced as "*the* answer"; rather, it is intended as a starting point for allowing various stakeholders to identify what they believe is right about the architectural approach and what needs to be changed (or whether the suggested architectural approach is even the right one to pursue at all).

The architecture presented addresses two levels of interoperability. The first level discusses how nodes or "stations" within a telemedicine system can be composed and how the resources within a station federate to deliver its functionality. The second level considers how different stations in a system discover each other's existence and then begin transacting business. Within each of these levels are a number of features that support different aspects of interoperability. As will be discussed in the next chapter, a key goal in laying out this architecture is to ensure that vendors are able to move towards full compliance in stages, as their business environment dictates, and are not constrained to implement portions of the architecture that do not support their business objectives.

## *3.1 Overview*

The proposed station-level architecture is built around three sets of interfaces (Figure 22). The station-to-device interfaces include those interfaces that enable medical instruments, patient record cards, user interface components, etc. to be added to and removed from the station in plug-and-play fashion. The station-to-station interfaces include those mechanisms used to allow remotely located stations to discover and interact with each other. The station internal interfaces consist of those mechanisms needed to allow various station components to interact with each other. The main reason for addressing these station-internal interfaces explicitly (as opposed to leaving the details of these interfaces to each station manufacturer) is to allow for the creation of telemedicine stations from independently developed components, including those not originally designed for use in telemedicine applications.

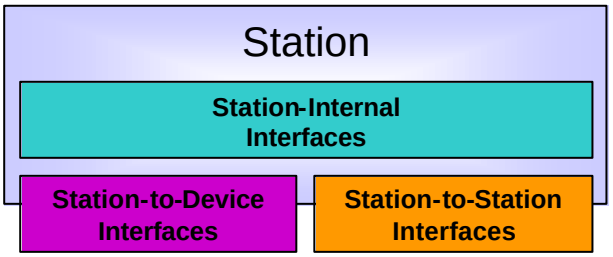


**Figure 22. Three Sets of Interfaces**

Using these three sets of interfaces, it is possible to create stations that support a variety of telemedicine concepts. For example, Figure 23 is a monolithic, stand-alone station

design that supports plug-and-play operation of some number of devices. Examples of this kind of device might include a homecare unit that contains an integral display and controls and that allows a patient using the device to plug in a blood pressure cuff, a thermometer, etc. as needed. It could also be a PDA capable of local area wireless communications (e.g., Bluetooth or infrared) that interacts with medical instruments that are carried around by the PDA's owner or that are dynamically discovered at each bed as the caregiver moves from room to room. Note that in both of these cases, the station developer has complete freedom with respect to how the station is implemented and is bound only by the station-to-device interfaces for those devices that he cares to support.

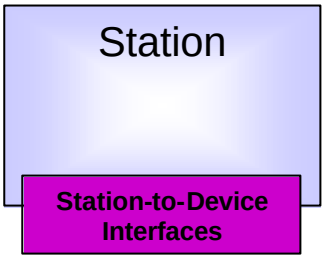


**Figure 23. Pattern for Plug-and-Play Devices**

Figure 24 is the pattern that would be used if the developer of a station cared only about making sure that his station could interoperate with those of other vendors. As in the previous example, a station of this sort could be implemented in anyway that the developer wished just as long as those the design complies with those station-to-station services that the station supports. This might include person-to-person communications or it might not. It might include allowing remote stations to access locally stored records. It might include enabling remote stations to "lease" and control local medical devices. This design pattern allows for the use of proprietary station-to-device protocols and says nothing about whether the station is rendered as a single monolithic unit or as a suite of interacting components.

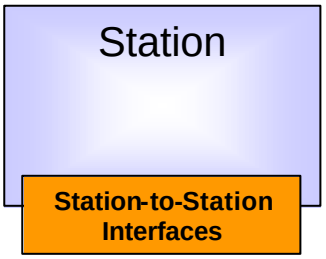


**Figure 24. Pattern for Interoperable Station**

The final basic design pattern is the distributed station, as shown in Figure 25. In this design, a station can be dynamically assembled from a suite of available components. In a home environment, a station of this sort might consist of a "base station" to which some number of medical devices are attached, a computer-enabled television, a network storage device on which the family's personal medical records are stored, and a home

network that ties all of these components together. On the battlefield, one distributed system of this sort might consist of the human interface device and computing employed by a medic and the instrumentation, records, and computing resources integral to the battle gear worn by the soldier that the medic is treating.

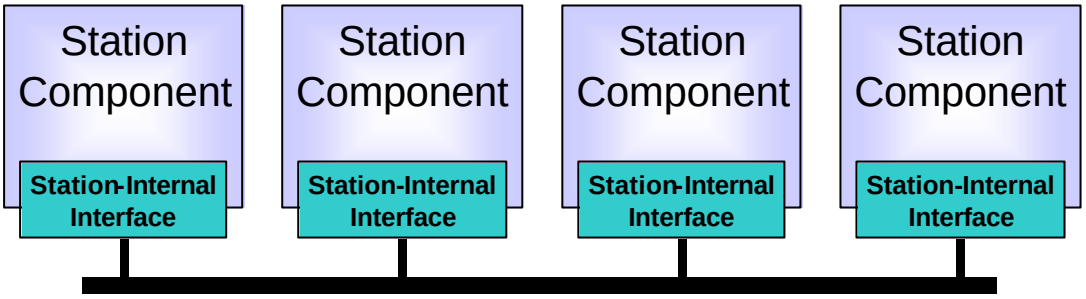


**Figure 25. Pattern for a Distributed Station**

Combining these patterns in different ways yields other interesting combinations. For example, the base station in the networked home care station just described might be replaced with a base station that supports plug-and-play addition of medical instruments and a "gateway" device that enables wide-area access might be added to enable this station to now interact with others (Figure 26). This final level of interoperability show be the ultimate goal of the telemedicine community in as much as it will enable a tremendous amount of innovation and should remove many of the key technology barriers related to guaranteeing "anywhere, anytime" access to healthcare. At the same time, reaching this ultimate goal will take time with the exact path to follow necessarily being driven by market needs and regulatory dictates.

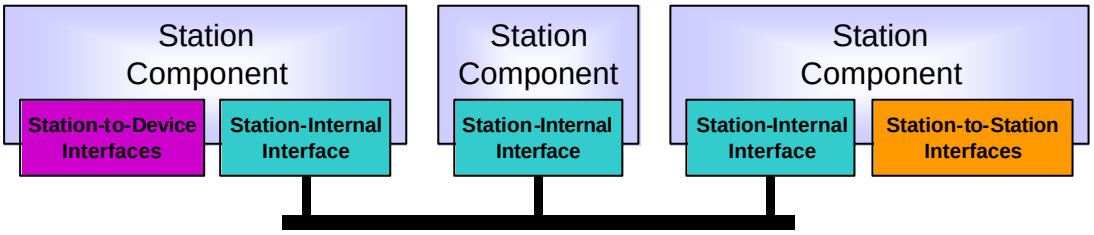


**Figure 26. Distributed, Interoperable Station Supporting Plug-and-Play**

Figure 27 presents a notional architecture for a telemedicine station that support plug-and-play operation of the various components from which it is composed. As noted in the last chapter, a central goal of this design is to allowed ad hoc assembly of systems.

At the heart of this design are an ***internal communications bus*** and a ***station registry***. Whereas most of the other components within the design are generators or consumers of information, the internal communications bus represents those mechanisms within the

station used to move this information between the components. Depending on the interoperability objectives being pursued, this bus might implemented using:

- a local network that support device-to-device messaging
- an operating system that facilitates interprocess communication
- a monolithic software program whose various pieces communicate via "function calls" and shared memory structures, or
- (most likely) some combination of the three.

A key feature of this bus is its ability to support quality of service requirements of the rest of the components that make up the station (e.g., it can deliver data from a medical device to a given user interface component in a specified amount of time). The station registry is a collection of data and associated software that is used by the station to catalog the list of components that the station can offer to support various clinical functions.

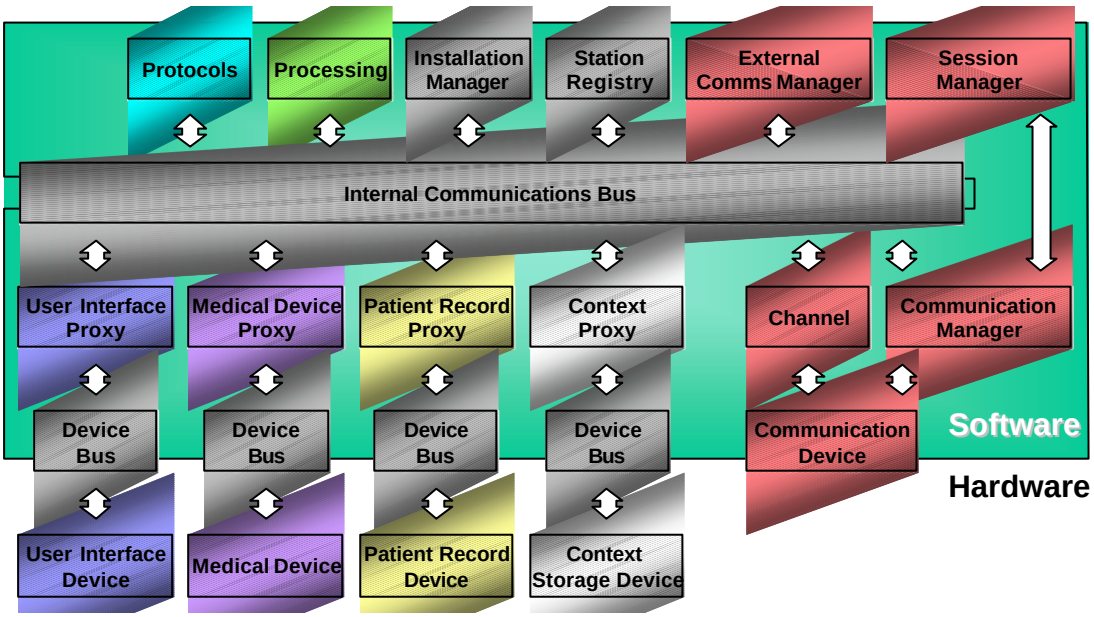


**Figure 27. Logical Station Architecture**

The components at the bottom of the diagram represent *physical* resources that may be added to or removed from a station. These include ***user interface devices***, ***medical devices***, ***patient record storage devices***, ***context storage devices***, and ***communication devices***. In the case of user interface devices, this can include things like:

- the input and display devices that are native to a computer on which some portion of the station software runs
- an application running on a PDA
- a web-page running on an "enhanced TV"

Medical devices can include both traditional diagnostic and therapeutic instruments as well as non-traditional elements (e.g., a load cell on a bed that tells how often a person is getting up each night) that help provide a bigger picture of a patient's well being. Patient record storage devices are included in the design as pluggable components to

accommodate the existence of patient cards, network-ready disk drives, etc. Communication devices can include things like the modem or Ethernet card in a PC that the station uses or a gateway device on a network that serves as the station's internal bus.

For each of the first three devices types, Figure 27 shows the devices connect to the rest of the station by means of a ***device bus***. This part of the station exists to support "hot plugging" of the devices into the station. The ***proxies*** that are shown between the internal communications bus and the device buses exist to encapsulate vendor-specific device communication behind standard ***interfaces*** that are understood by other system resources.

In order to shield station components from the details of what kinds of devices exist to support external communications and how these devices are controlled, a ***communications manager*** is provided. Upon request, this software component establishes ***channels*** through which a station's components can connect to external resources. An ***external communications manager*** provides a single point of contact for mediating whether sessions will be established with remote stations. ***Session managers*** provide a means for external systems to "lease" a station's resources and, working with the communications manager, also monitor the status of external communication networks to ensure that quality of service requirements associated with interacting with remote resources are being met.

***Contexts*** store information regarding patient and caregiver preferences and are responsible for configuring a station (or stations) in accordance with these preferences whenever a user logs onto a station. Every station will have a standard configuration that it uses in the absence of other configuration information. This is referred to as the ***default context***. ***Caregiver contexts*** and ***patient contexts*** can be introduced to the station via some form of ***context storage device***.

Most of a station's higher level "intelligence will be vested in three groups of components (one of which is the station's ***contexts***). The ***processing*** block in the diagram represents the data transformation components and other specialized building blocks (e.g., statistical analysis tools, "intelligent agents", person-to-person communication clients) that reside on the station. ***Protocols*** are components that are responsible for intelligently acquiring resources needed to implement a clinical function, instructing these resources on how they are to interoperate, and then monitoring the operation of these resources for key events (e.g., a "leased" device being removed from the station prematurely or the operation terminating under normal conditions).

Finally, in order to allow for automated downloading and installation of components onto a station, the ***installation manager*** is included in the station architecture. This component, working in conjunction with the device bus managers and the registry, determines when a device that has just been added to station lacks the necessary infrastructure (i.e., proxy, processing, protocol, patient record, and user interface software) to function on that station and, when permitted by the station's operator, downloads from the web and installs the software that provides this infrastructure for the device.

Figure 28 depicts the notional system-level interoperability architecture. At this level, the goal is to allow independently designed and implemented systems to locate each other, explore each other's capabilities (subject to each station's access control rules), to negotiate with each other and with the networks that they will use to determine how a given session will be run (e.g., what QOS requirements will be levied and what resources will be leased from each other), and to then conduct collaborative operations. Key architectural components at this level include ***telemedicine stations***, ***registry servers***, and ***networks***. Stations are the collections of resources described in the first part of this section. Registry servers are computer-based systems that allow stations to advertise their location and, if desired, services and that allow stations to locate either specific stations or stations that offer particular kinds of services. Networks consist of those telecommunication components that carry information back and forth between stations in a telemedicine system.

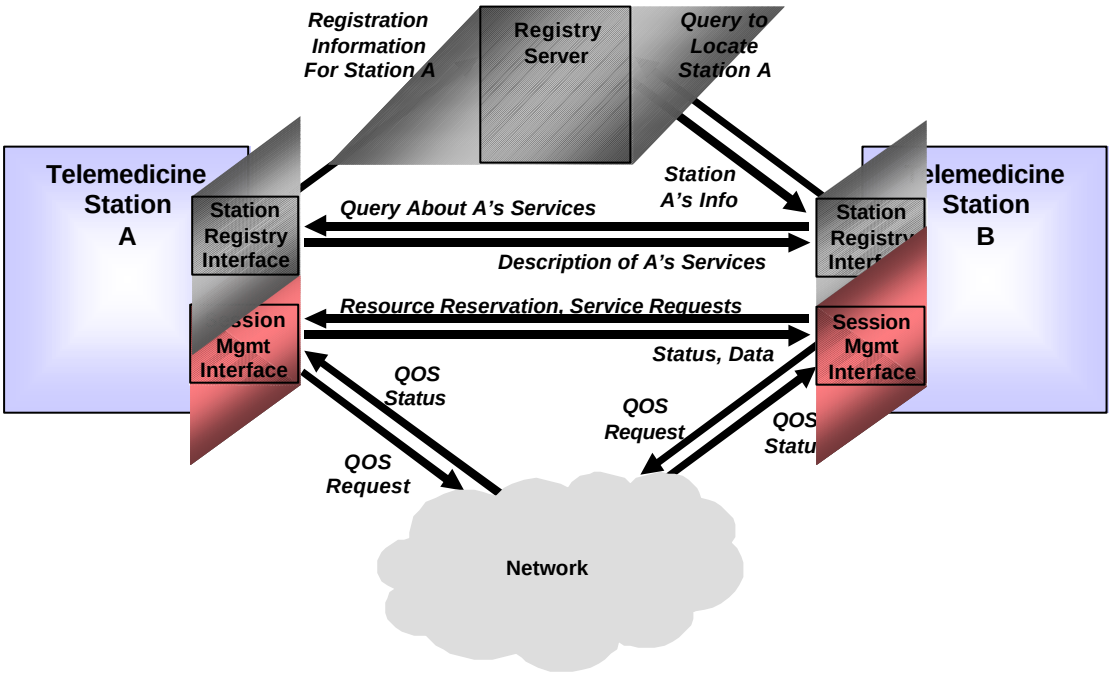


**Figure 28. Station-to-Station Interoperability Architecture**

At this level, stations present two interfaces to other stations. The ***station registry interface*** provides a way for one station to explore the capabilities that another station can provide. These capabilities can include any of the resources discussed in the station-level architecture (medical devices, user interface devices, patient records, processing components, and even protocols and communications). The ***session management interface*** provides a means for stations to "lease' these services for a period of time and for the quality of service components in a network to negotiate a contract with a station and to report, as needed, of the status of that contract.

Figure 29 illustrates the final relationship within the overall system design – that part used for automatic configuration of a station. When a new device registers with the station, the registration event is passed to the installation manager which checks to see if all of the software needed to support the new device is present. If not, the installation manager queries the device (or external server) to learn where the supporting software is stored. The answer is returned as a URL. Sending a request to this site, the installation manager receives back a package of one or more components, which it unwraps and then installs each of the constituent components on the station. On command from the installation manager, each of the components then registers itself with the station and, when this process is complete, the device is ready for use on the station. As a final step, the package may indicate a given protocol for the installation manager to automatically execute in order to support activities like running software or streaming videos that demonstrate how to use the newly installed device.

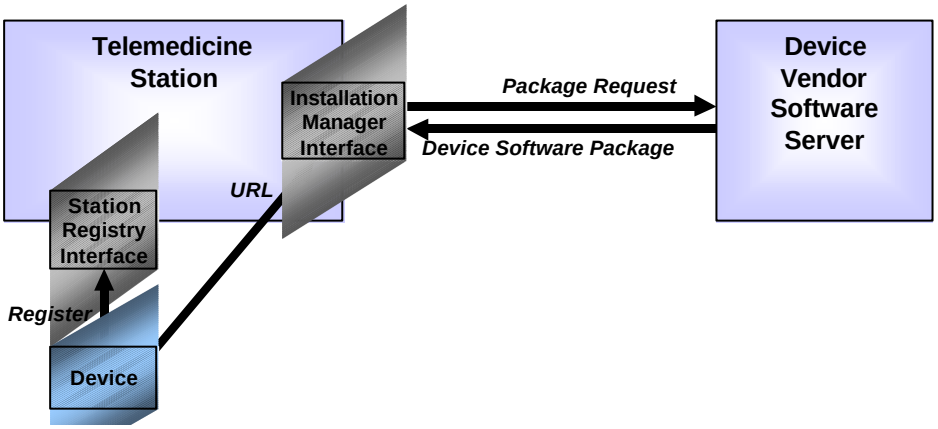


**Figure 29. Self-Configuration Architecture**

## 3.2 Architectural Details

Given this overview we now come to the question of how to make our systems function in the way we have described. In proposing how to address this question, we will begin by describing an example of how the various parts of a system operate. Using this example as context for our discussion, we will look at the patterns that constitute the foundation of this architecture and then examine how these patterns express themselves in each of the various component types in the architecture.

### 3.2.1 An Example

As an illustration of how the architecture is to work, consider the simple system shown in Figure 30. The ***user interface framework*** represents the top-level user interface constructs (menus, main buttons, etc.) used to initiate functions in the system. The ***pulse oximeter user interface module*** is a user interface component that is presented when a user needs to control a pulse oximeter and to display the data generated by the ***pulse oximeter***. In this scenario, the pulse oximeter accepts start and stop commands and outputs a continuous oxygen saturation waveform. The ***waveform statistical analyzer*** accepts waveform data and generates a variety of statistics that characterize the waveform

(e.g., average, max, min). The ***pulse oximeter operation button (and event handler)*** represents that user interface control contained in the user interface framework that is used to start and stop the operation of the ***pulse oximeter protocol***. This protocol contains instructions regarding the kinds of components that are needed to support the protocol's operation, the ways in which these components need to be interconnected, and events that are to be monitored during the time that the protocol is active. The ***registry*** exists to allow components to discover each other's existence.

To start, assume that the three resources in the left of the diagram have just been added to a station. Their first action is to register themselves with the station (Figure 30). Then,

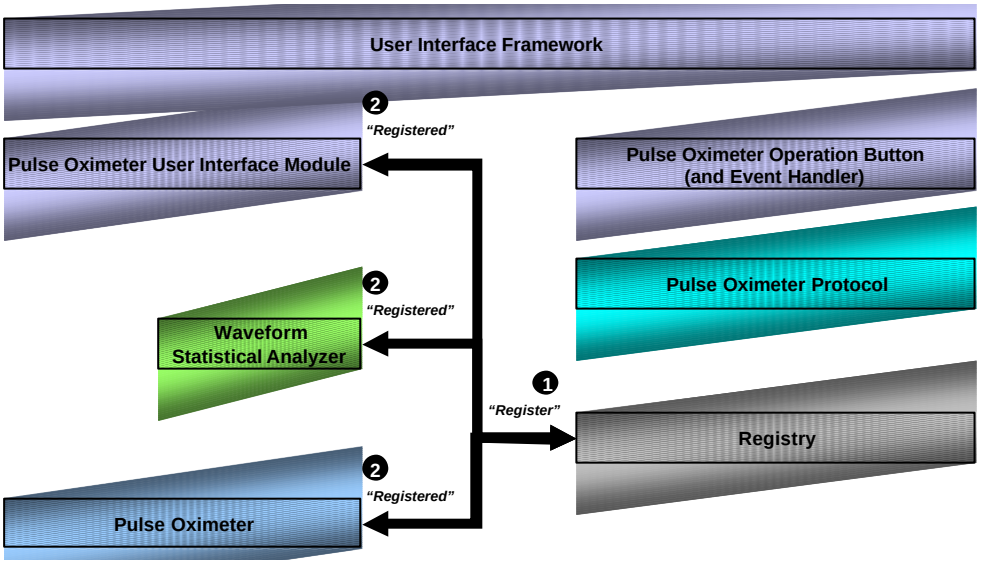


**Figure 30. Registering Components**

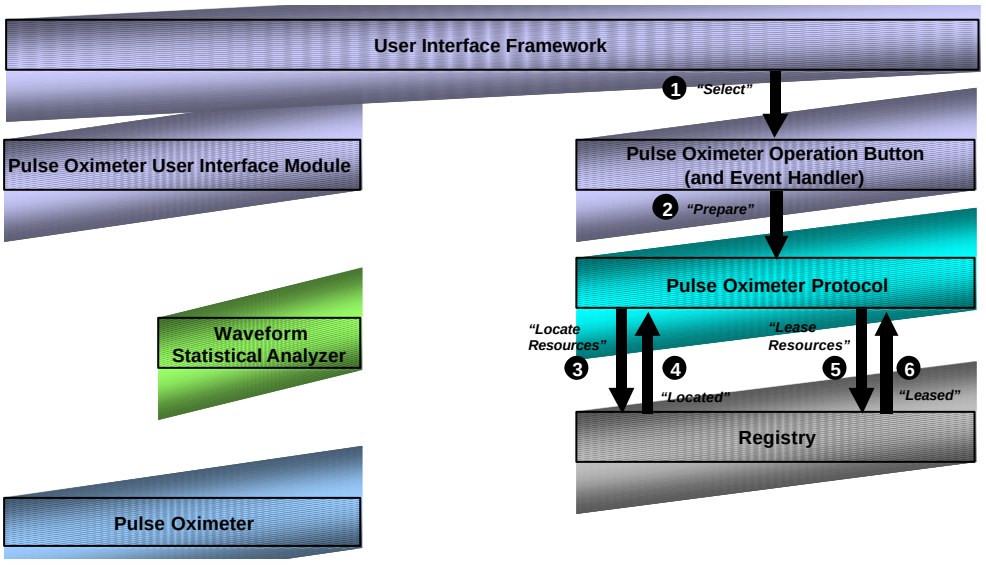


**Figure 31. Initiating a Protocol**

when the user selects the pulse oximeter operation from the user interface framework (Figure 31), the associated event handler instructs the protocol to "prepare" itself by leasing needed resources from the registry and instructing each of the leased components to subscribe to specific services offered by other leased components (Figure 32). Once done, the protocol notifies the event handler which instructs the UI framework to display the pulse oximeter user interface module for the station's operator (Figure 33).

System operation now proceeds with the user instructing the station to start (or to stop) taking pulse oximeter readings. While it is operating, the pulse oximeter sends its waveform data to its subscribing components (i.e., the pulse oximeter user interface

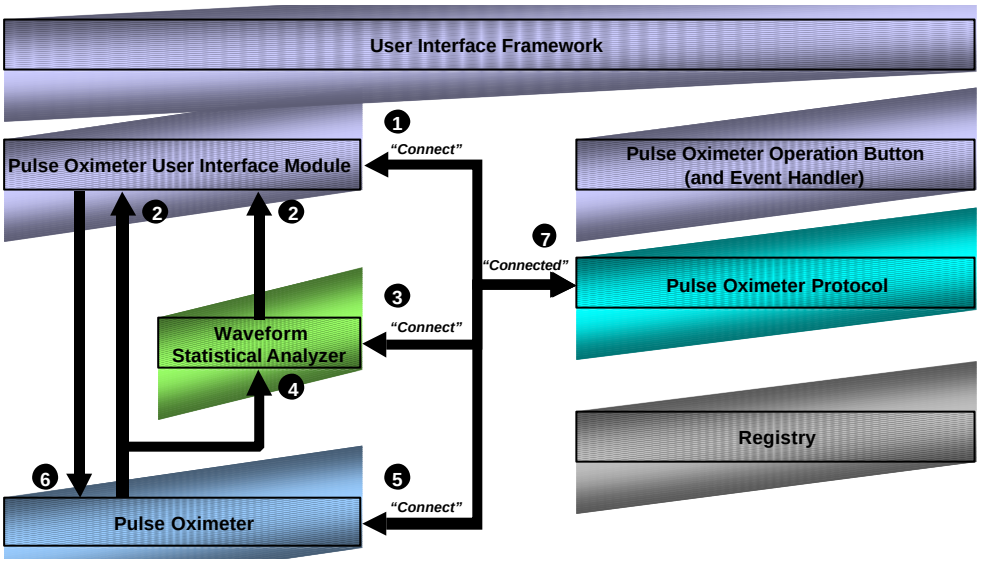


**Figure 32. Connecting Components**

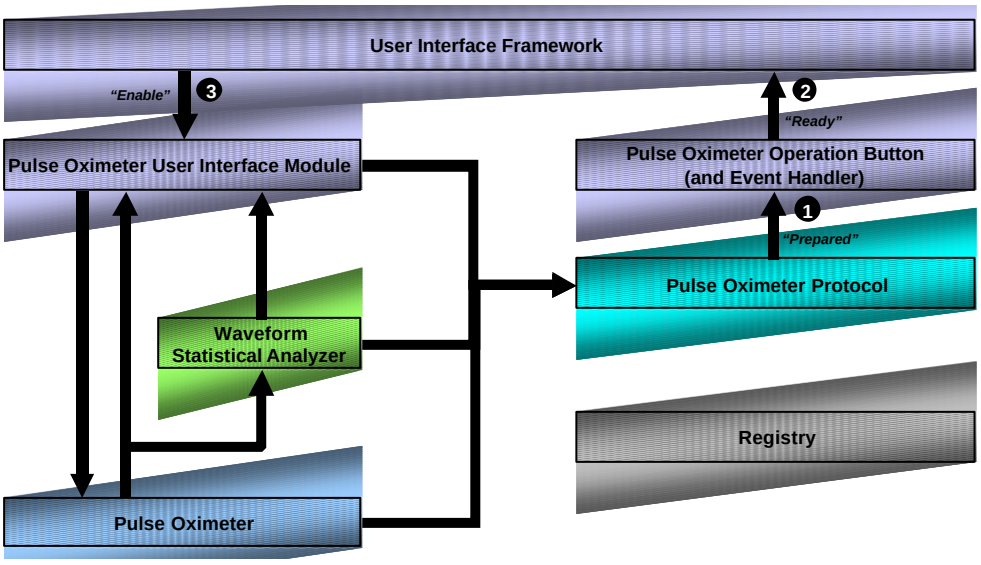


**Figure 33. Enabling User Operation**

module and the waveform statistical analyzer) and the analyzer sends the data on to its subscriber (Figure 34). If, during the operation of this network of components, any of these components experience an event that compromises its ability to support the protocol (e.g., the pulse oximeter is removed from the station), then the affected components notify the protocol which must then decide how to handle this situation.

When finished with the pulse oximeter, the user may "deselect" the device on the user interface framework, which results in the pulse oximeter user interface module being disabled and the pulse oximeter protocol being told to terminate itself (Figure 35). In

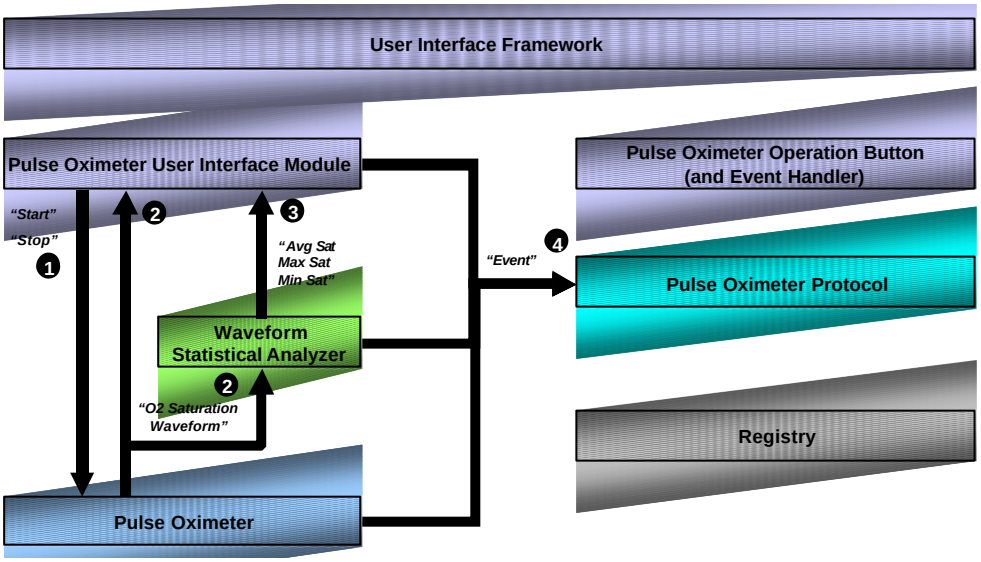


**Figure 34. Using the Component**

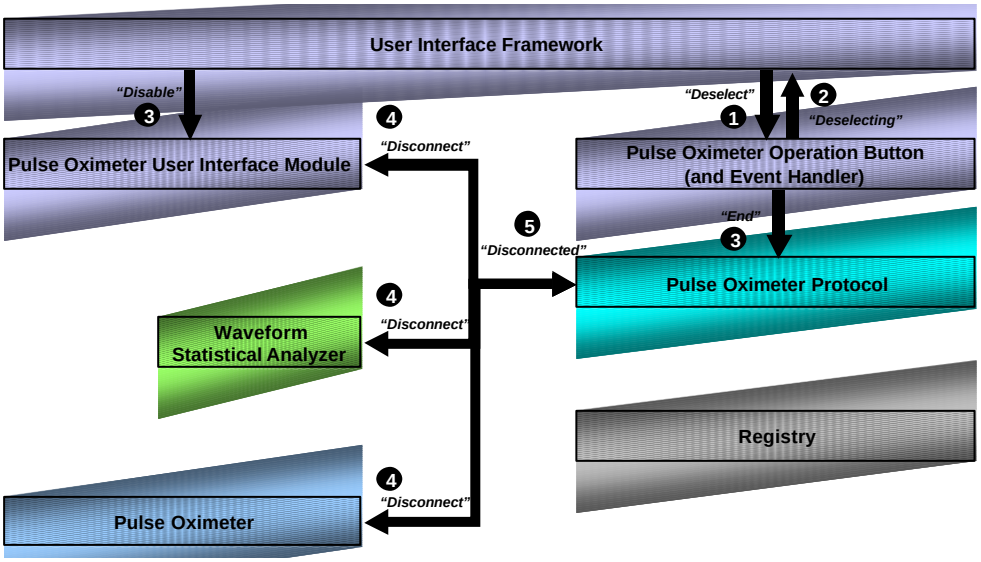


**Figure 35. Disabling User Operation and Disconnecting Components**

turn, the protocol instructs each of the leased components to terminate its lease. The protocol then notifies the registry that it is vacating its lease on these components and tells the user interface event handler that it is ending (Figure 36). The event handler then passes this fact on to the user interface framework that returns its interface to the same state that it was in before the pulse oximeter was first selected.

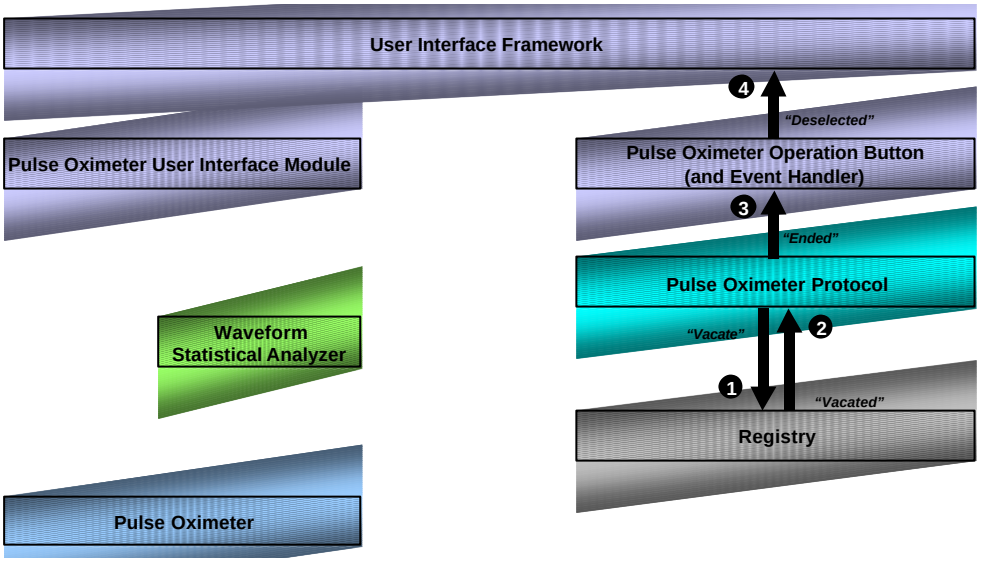


**Figure 36. Vacating the Leases**

### 3.2.2 The Heart of the Architecture

As illustrated in the previous example, the architecture is built on a core set of capabilities and a handful of key component structures. The core capabilities supported by components in the architecture are:

- registration,
- leasing, and
- subscription.

The key components are:

- the generic component,
- the registry,
- protocols,
- user and caregiver contexts,
- the internal communications bus, and
- device buses and factories and proxies.

Every component within a station presents the station with a set of services. These services are accessed through ***interfaces*** that the component implements. These services include both the core capabilities mentioned above (note that the "leasing" capability is peculiar to the "protocol" components) and the component-unique capabilities that constitute the component's reason for being (e.g., those services that make the component

a particular kind of medical device or a processing module or a user interface element). In this sense, every component in the system is both like all of the others (in as much as it implements a set of functions that are common to all devices in the architecture) and different from all others (in as much as it implements a set of services unique to that component's type and function).

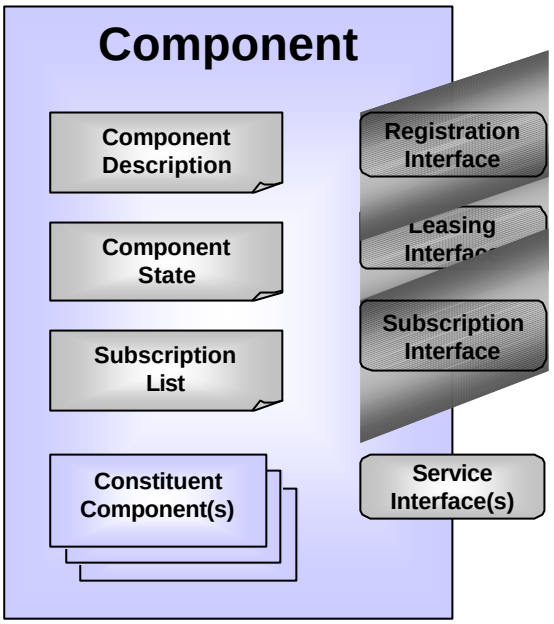


**Figure 37. Structure of the Generic Component**

Figure 37 presents a notional view of the generic component – the prototype from which all other components are derived. In addition to the three core interfaces, a component will contain three sets of data: the component description, the component state, and a subscription list. The component description provides a component with a means of characterizing itself for other components. The component state consists of those variables reflecting the component's internal state that can be monitored by other components. The subscription list identifies external components that have subscribed to messages generated by the service interface and by internal state changes. As a final element of the component's structure, a component may itself contain other components (e.g., a defibrillator will contain both a cardiac monitor and a power discharge unit that act both independently as self-contained entities and interdependently as a defibrillator system). Depending on the nature of the component, these constituent components may be treated as independent components or addressed through the "root" component.

### *3.2.2.1 Core Services*

#### 3.2.2.1.1 Registration

One of the first steps that a component takes when being added to a station is to register itself with the station. This begins with the component broadcasting an announcement of its existence to the rest of the components within the station. The broadcast

announcement contains the component's address (this may be fixed or may be assigned dynamically each time the component is added to the station), which is then used by the registry in all subsequent communications with the component. The announcement also contains the component's globally unique ID (a combination of manufacturer, model, and serial number). In response to this broadcast announcement, the registry provides the component with the registry's address so that all subsequent communications can continue in a directed fashion. It also provides a registry-unique ID to the component that the component can use as a shorthand way of referring to itself when communicating with the registry (Figure 38).

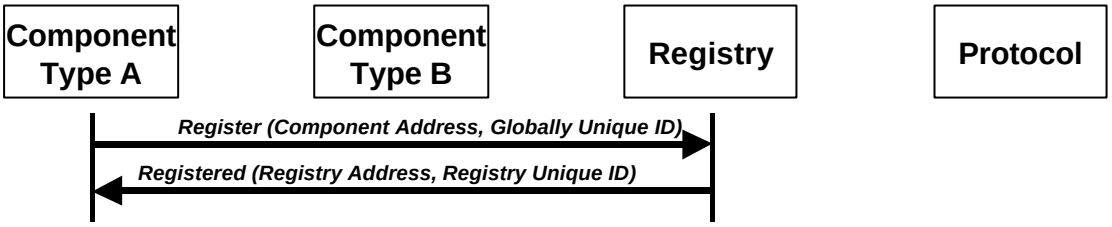


**Figure 38. Announcing the Component's Presence**

Components that are added to a station can be "installed" in one of two ways: "temporarily" or "permanently". A component that installed temporarily is known by the station only as long as it is attached to the station. For example, a stethoscope carried by a doctor making his rounds will be installed by the station in each room that he enters and will be available for use as part of that room's station for as long as the doctor remains there. When the doctor leaves a room and that room's station is no longer able to communicate with the stethoscope, the stethoscope information is removed from that station's registry and the station "forgets" about that device's existence.

When a device is "permanently" installed on a station, all of the information about that device that has been stored in the station's registry is retained when the device is physically disconnected from the station. This enables a remote station to determine that a station being queried has the ability to support certain device-dependent operations even when these devices are not attached to the station. For example, this would allow a home care nurse conducting a tele-visit with a new patient to see that the patient's station is able to take a temperature even though a thermometer is not attached to the patient's station at the time that the exploration of station capabilities is being conducted.

When a device is temporarily installed (or a "permanent" device is being attached for the first time), it goes through the registration process show in Figure 39. In this process, the registry issues a sets of queries aimed at fully discovering the nature of the component. This includes discovering more about the component itself (e.g., "Does it want to be installed on a "permanent" or "temporary" basis component or "Does it have a "common name" that a user would use to find it?"), about which interfaces the component supports, about which elements of the component's internal state can be monitored, and about the existence of constituent components within the component.

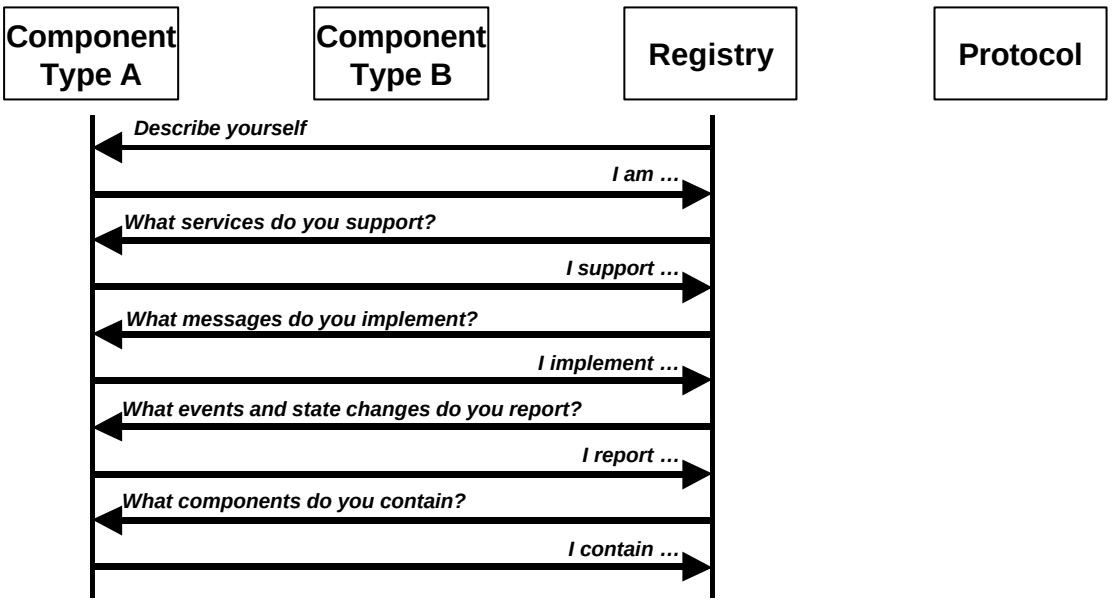


**Figure 39. Registering the Component's Description**

Once a device is registered, it may be unregistered in several ways. First, if a component has been installed on a temporary basis and is not currently "leased" (i.e., reserved for use by other devices, as described below), then removing it from the station results in the registry eliminating all information about that component. Second, if a temporary device is removed while still leased by a "protocol" on that station, then the registry queries the leasing protocol to determine if it is okay unregister the component (Figure 40). If the leasing protocol vacates its lease, then the device description stored in the registry is eliminated. If the protocol chooses to not vacate the lease, then the registry simply notes that the device is not attached to the station. When the device is once more attached to the station, it announces its presence (as described above) and the registry handles it like a permanently attached device (i.e., notes its new address if one has been assigned and then returns it a message that includes the registry location and the registry-unique ID assigned to the component). If a leasing protocol that has indicated that it does not want to vacate its lease later decides (before the device is reattached) that it wants to vacate the

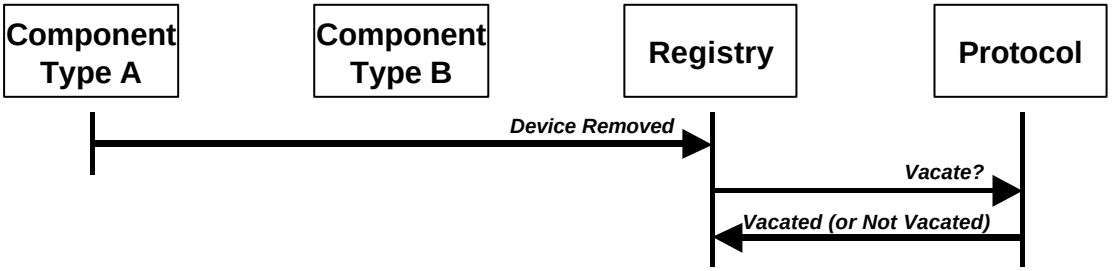


**Figure 40. Removing a "Temporary" Component That Is Currently Leased**

lease, it sends a message to this effect to the registry which then clears the device description from the registry.

If a permanently attached component that is not currently leased is removed from a station, the station's registry notes that the device is no longer present but does not delete the device's description. If the component is currently leased when the removal occurs, then the registry queries the leasing protocol to determine if the protocol wishes to vacate its lease. If so, then the registry notes that the lease has been vacated and that the device is no longer attached. If not, then the registry simply notes that the device is no longer present. In both cases, the device description remains intact.

Finally, when it is necessary to remove the description of a component that is permanently installed on a station, either of two methods will be supported. In the first, the component itself can request that its registration be terminated (Figure 41).

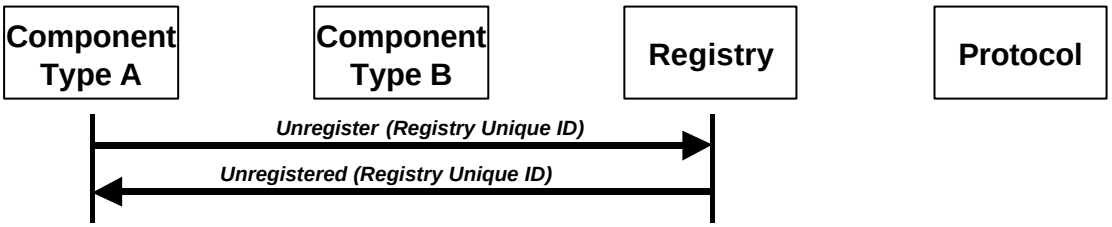


**Figure 41. A Component Requesting That Its Registration Be Terminated**

Alternatively, each station will provide a suite of utilities used to administer the station. Among other operations supported, these utilities will allow an operator to inspect the internals of station's registry and to either remove from the registry a given permanent device description or change the "persistence" attribute of a given device from "permanent" to "temporary" and vice versa (Figure 42).

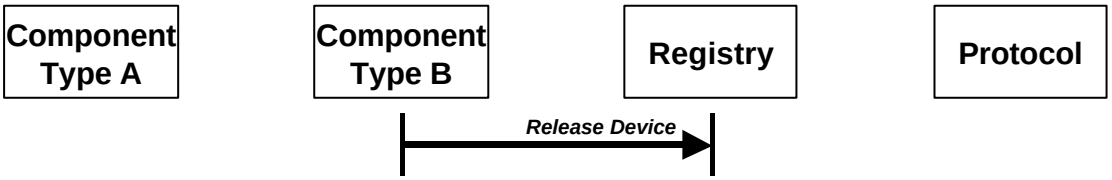


**Figure 42. Preparing to Remove a "Permanent" Component**

#### 3.2.2.1.2 Leasing

Once registered, components are available for leasing by protocols. In establishing a lease, a protocol is given the right to control certain aspects of a leased component's operation, such as specifying to which other components the leased component is to attach itself for the purposes of sending and receiving data and commands. The goal of leasing is to provide a clean way of dealing with resource contentions in environments in which station resources (devices, medical records, etc.) might be shared among multiple protocols or even among multiple stations. Of particular concern here is avoiding

situations in which components are "fighting" to control a given resource's operation or in which the nature of data being generated by a given component becomes confused due to multiple components trying to "write" to the same data storage location in a component. Depending on the nature of the particular component element being leased, a lease may be "exclusive" (i.e., only one protocol may establish a lease at a given time) or "shared" (i.e., more than one protocol may establish a lease on the specified element at the same time). In the case of shared leases, the number of leases that can be established may be "unlimited" or "fixed", as specified by the component in question.

To establish a lease, a protocol sends either of two types of queries to the registry (Figure 43). In the first, the protocol requests that it be allowed to lease services from a specific component. In the second, it requests the opportunity to lease services from a specific *type* of component. In response, the registry returns the ID(s) of the component(s) matching the query along with the associated lease status(es) of the requested services on the selected component(s) . If needed, the protocol can explore one or more of the components in more detail to determine whether any of them meet its requirements. When satisfied, the protocol then selects a component that currently has leasable resources of the type desired. It then passes a request to lease these resources to the registry. In turn, the registry notes within its component description which resources have been leased and by which protocol. At this point, the leased resources are ready for use by the protocol.

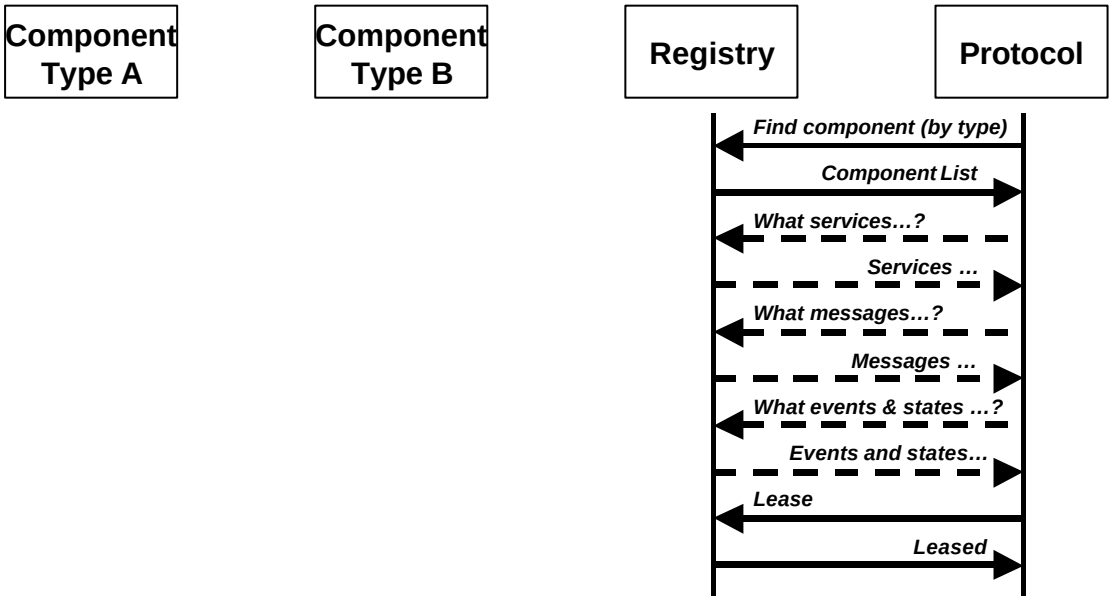


**Figure 43. Leasing a Vacant Component**

At times, it may be that the components and associated resources that a protocol wishes to lease are already claimed by other protocols. In this case, the protocol can direct the registry to query the other leaseholders to see if any of them are willing to vacate their leases. If one or more are willing, the registry returns a list of component IDs and lease

statuses as before and the protocol then establishes leases with the desired component (Figure 44). If none of the protocols are willing to vacate the needed leases, then the requesting protocol can direct the registry to terminate the leases held on a specified component and resources. While this action still may not result in vacation of the specified lease (in which case the protocol notifies the user of its inability to execute due to its inability to acquire certain resources), the most likely results of this directive (as this will most likely be done under user control) will be the early but orderly termination of whichever protocol has leased the terminated resource. The protocol sequence used to achieve this will be essentially the same as that shown in Figure 44 and will differ only in the fact that the request to vacate that is sent to protocols leasing the desired resources applicable components is replaced with a directive to vacate that is sent to a specific protocol.

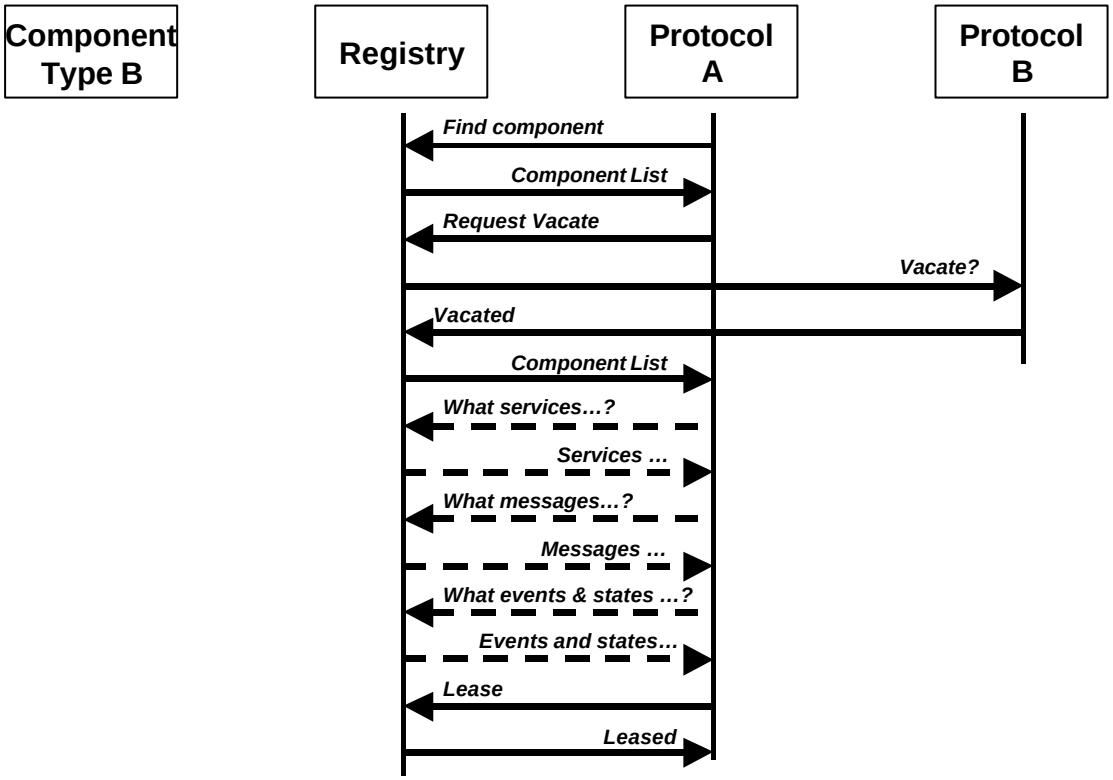


**Figure 44. Requesting a Lease on an Already Leased Component**

Under normal circumstances, when a protocol has run its course or is externally terminated, it releases the component resources that it has leased back to the registry for reuse by other protocols. It does this by sending one or more "vacated" message to the registry (depending on how many components and resources it has leased).

In leasing and vacating resources, a protocol may pursue either of two strategies. In the first, it may seek to acquire all of the resources that it will need in the course of it operation and then not release these resources until it is ready to terminate its operation.

Alternatively, a protocol may acquire and retain only those resources that it needs for the moment. If multiple resources are required sequentially (rather than simultaneously) during the course of the protocol's operation, it may go through several iterations of acquiring, using, and then releasing resources. For example, a protocol might be created that exists to take vital signs readings. It starts out by acquiring the blood pressure monitor, uses this device, and then releases it back to the station. Next, it acquires, uses, and releases a thermometer, and so on.

In leasing a component, the use of "persistence" in the registry (i.e., marking some devices as "temporary" and some as "permanent") means that even though a given component is listed as available for leasing, it may not currently be attached to the station. In these cases, the leasing protocol must be able to direct the user to attach the specified resource to the station and to then, working in conjunction with the registry, to determine that this has occurred.

Finally, it should be noted that, during leasing, a protocol will acquire two broad classes of resources: those that generate or consume data and those that are used to transport data between the generators and consumers. In leasing the first class of component, one of the concerns that a protocol must address is the volume of data that the generators will produce when configured as desired and the ability of the consumers to absorb this data. In addition, the managing protocol must also ensure that the communication "channels" that it acquires for interconnecting the consumers and generators can support the required data rates of these components and services.

#### 3.2.2.1.3 Subscription

Once leased, a component in the system has the potential to generate some number of messages either in response to messages that it has itself received or in response to changes in the component's internal state. Each such message can be sent to one or more other components in the system. The list of "subscribers" to which a given message is to be sent is established by components sending subscription requests to the component that generates the message, asking that they be added to the list of components that receive the message. Each such subscription remains in effect until canceled by the subscribing component or until the lease on the component is vacated (in which case all active subscriptions are terminated by the leased component).

Once a protocol has established leases on a component's resources, it is free to begin connecting those resources to the resources of other components that it has leased. This subscription process proceeds with the protocol issuing a series of commands to each of the leased components instructing them to subscribe to specific messages generated by specific components and to use specific communication resources in support of each subscription (Figure 45). As shown in the figure, Component A is instructed to subscribe to a specific message generated by Component B. In response, Component B registers this subscription in its list of subscribers and responds with an acknowledgement to Component A. Component A then notifies the controlling protocol that it has successfully established a subscription. Using the subscription ID returned from Component A, the protocol then instructs Components A & B to use a specific channel on the internal communications bus when sending and receiving the message associated

with the subscription (see description of the Internal Communications Bus for information on how channels are acquired and released).

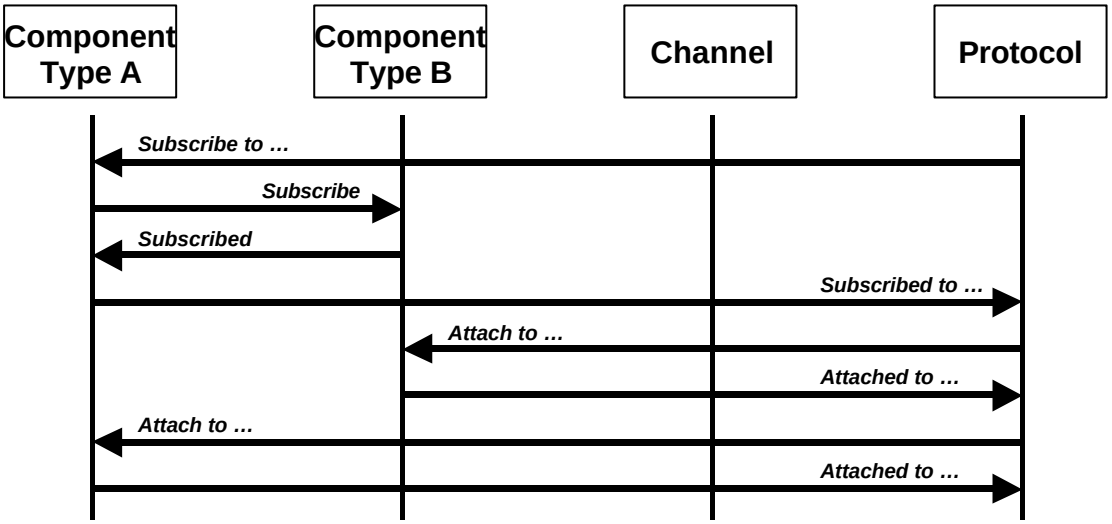


**Figure 45. Protocol Directing Establishment of a Subscription**

In establishing subscriptions, a protocol can act like any other component by itself subscribing to messages generated by the components that it has leased. This allows the protocol to then watch for key events, such as an operator configuring a device from its front panel, which might require intervention by the protocol.

While it was implied earlier, it is worth stating more directly that one of the goals of the reference architecture is to allow for the creation of "virtual" stations using components physically attached to different stations. While this will typically be used for allowing a given station to subscribe to the medical instruments on a remote station, it is also the mechanism that will be used for allowing protocols running on a remote station to be accessed from a local station and used to control the resources on the local station (e.g., to allow a caregiver to carry his "work environment" with him). In support of this, the architecture supports an extended addressing scheme that allows for components to be identified using both their registry ID (the unique ID used within a station to refer to a registered component) and the ID of the station of which they are a part. As with intra-station subscriptions, the ability to subscribe to a remote component's messages presumes that the protocol has been able to acquire an adequate communications channel to the external component in order to support this subscription.

As a protocol terminates, it instruct each leased components to cancel all of the subscriptions that have been placed with it. In like fashion, the protocol instructs any channels associated with these subscriptions to terminate their attachments. As components clear their list of subscribers and channels clear their list of attachments, they notify the protocol of this fact (Figure 46).

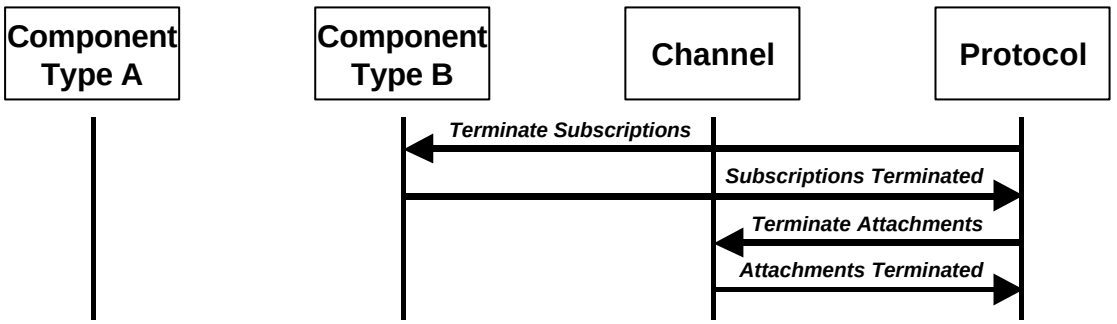


**Figure 46. Protocol Directing Termination of a Subscription**

### *3.2.2.2 Key Components*

Given these core interactions, the next question to address is structure of the components that make up the Telemedicine Reference Architecture. As with the interactions, this section provides an overview of the topic.

#### 3.2.2.2.1 The Generic Component

The generic component is meant to be a pattern/template from which all other types of components in the architecture are derived. Specific component types, such as medical devices, which are discussed later will implement the features described here and extend these features with their own component-specific features.

The generic component stores three sets of data and, possibly, one or more constituent components, which themselves contain the same kind of structures as the generic component. The data sets are the "component description", the "component state", and the "subscriber list". In the descriptions below , items contained in braces (e.g., ({XYZ}) indicates a variable length list of items of type XYZ.

| **Component Description** | |
|---|---|
| Registry Component ID | Unique ID assigned by registry |
| Registry Location | Address of the registry |
| Component Type | = Class + Type |
| Class | Identifies component's general type |
| Type | Uniquely identifies component's function |
| Component ID | = Device Manufacturer + Model + Serial # |
| Device Manufacturer | Uniquely identifies the device's manufacturer |
| Model | Unique within the manufacturer's product line |
| Serial Number | Unique within products of a given model |
| User-assigned Name | Plain-text name assigned by owner/admin |
| Component Location | "Base address" of the device |
| Component Persistence | Indicates if should be treated as "permanent" |
| Constituent Component List | = {Constituent Address} |
| Constituent Address | Relative address for constituent component |
| Security-related Parameters<br>Service List | Tree listing component's security capabilities<br>={Service Type+Service Location+Msg List} |

| | |
|---|---|
| Service Type | Specific set of capabilities component offers |
| Service Location | Address of this set of capabilities |
| Message List | ={Msg ID+Direction+Content+Encoding+Prot} |
| Message ID | Unique within scope of component |
| Direction | Indicates if message is sent or received |
| Content | Variable/parameters contained in message |
| Encoding | Structure of specific fields & entire message |
| Protocol | Indicates protocol used to communicate |
| Message Location | Used to override Service Location, if desired |
| Lease Type | Indicates “exclusive” or “shared” lease |
| Share Count | If “shared”, gives number of shares allowed |

Each component will have a set of state variables that reflect the internal status of the component or that can be used to control the operation of the component. As appropriate, messages can be sent to a component to set certain of these variables. Likewise, for those variables that reflect component status, the component can generate status messages to subscribers when the values stored in these variables change.

| **Component State** | |
|---|---|
| Connection Status | Indicates if device is attached to station or not |
| Registration Status | Indicates progress to registration with station |
| Subscriber Count | How many subscriptions are currently set |
| Operating Mode | Operation \| Administration \| Calibration |
| Configuration Data | Device-unique configuration state data |
| Operating Parameters | Device-unique operating parameters |
| Alarms | Device-unique alarm parameters |
| Errors | Indicate device operation failures |

Subscribers to a component can include components that want to interact with specific parts of the component’s normal service interfaces as well as those that wish to receive status messages when the specific aspects of the component’s internal state change.

| **Subscriber List** | |
|---|---|
| Subscription ID | Unique for all subscriptions & all components |
| Subscriber | = Station ID + Registry Component ID |
| Station ID | Universally unique identifier for station |
| Registry Component ID | Unique ID assigned to subscriber by registry |
| Service Type | Specific set of capabilities of interest |
| Message ID | Specific message of interest within this set |

#### 3.2.2.2.2 The Registry

As with the generic component, the registry contains data elements, internal state, and services. Because the registry exists to support leasing of components within a system (and, secondarily, to provide persistent knowledge of a component’s membership in a station) much of the data contained in the components is also stored in the registry.

| Registry Description | |
|---|---|
| Station ID | Universally unique ID for station |
| Station Name | “Common name” given to the station |
| Station Type | Type of platform on which station is hosted |
| Registry Network Location | The registry’s global network address |
| Other (TBD) | Other information used by registry servers |

Internal state variables in the registry relate to key events in the registry’s lifecycle. These include:

| Registry State | |
|---|---|
| Registry Availability | Changes if registry becomes (un)available |
| Component Registration | Marks another device being added to registry |
| Registration Modification | Marks when device registration data changes |
| Termination of Registration | Indicates removal of component from registry |

Like other components, the registry maintains a subscriber list for tracking all of the components that subscribe to the various messages that the registry publishes. As with the generic component, each subscription contained in this list is characterized by the following data elements:

| Subscriber List | |
|---|---|
| Subscription ID | Unique for all subscriptions & all components |
| Subscriber | = Station ID + Registry Component ID |
| Station ID | Universally unique identifier for station |
| Registry Component ID | Unique ID assigned to subscriber by registry |
| Service Type | Specific set of capabilities of interest |
| Message ID | Specific message of interest within this set |

Each component registered will have its own component description which largely mirrors the data stored in the component itself.

| Component Description | |
|---|---|
| Registry Component ID | Unique ID assigned by registry |
| Registry Location | Address of the registry |
| Component Type | = Class + Type |
| Class | Identifies component’s general type |
| Type | Uniquely identifies component’s function |
| Component ID | = Device Manufacturer + Model + Serial # |
| Device Manufacturer | Uniquely identifies the device’s manufacturer |
| Model | Unique within the manufacturer’s product line |
| Serial Number | Unique within a products of a given model |
| User-assigned Name | Plain-text name assigned by owner/admin |
| Parent Component | *Registry Component ID* for containing device |

| | |
|---|---|
| Component Location | “Base address” of the device |
| Component Persistence | Indicates if should be treated as “permanent” |
| Component Presence | If “permanent”, indicates if attached to station |
| Constituent Component List | = {Registry Component ID of Constituent} |
| Constituent Component ID | ID for constituent component |
| Security-related Parameters | Tree listing component security capabilities |
| Service List | ={Service Type+Service Location+Msg List} |
| Service Type | Specific set of capabilities component offers |
| Service Location | Address of this set of capabilities |
| Message List | ={Msg ID+Direction+Content+Encoding+Prot} |
| Message ID | Unique within scope of component |
| Direction | Indicates if message is sent or received |
| Content | Variable/parameters contained in message |
| Encoding | Structure of specific fields & entire message |
| Protocol | Indicates protocol used to communicate |
| Message Location | Used to override Service Location, if desired |
| Lease Type | Indicates “exclusive” or “shared” lease |
| Share Count | If “shared”, gives number of shares allowed |

Leases established by various protocols are recorded using the following structure:

| **Component Leases** | |
|---|---|
| Lease ID | Unique for all leases on station |
| Lessee | = Station ID + Registry Component ID |
| Station ID | Universally unique identifier for station |
| Registry Component ID | Unique ID assigned to protocol by registry |
| Lease Description | |
| Station ID | Universally unique identifier for station |
| Registry Component ID | Unique ID assigned to leased component |
| Service Type | Specific set of capabilities of interest |
| Message ID | Specific message of interest within this set |

The following set of messages is used by a component wanting to determine to which station it belongs, by the registry to answer these queries, and by internal and external entities involved in establishing a station’s identity:

| **Station Messages** | |
|---|---|
| SetStationID | Assigns station universally unique ID |
| SetStationName | Station’s common (human readable) name |
| GetStationName | Sent to registry to retrieve naming parameters |
| StationName | Returned in response to GetStationName |

Notification messages generated for each of the various registry events include:

| **Registry Event Messages** |
|---|

| | |
|---|---|
| RegistryAvailable | Indicates registry has come online |
| RegistryUnavailable | Indicates registry has shut itself down |
| ComponentTypeAdded | Indicates given type of component registered |
| SpecificComponentAdded | Indicates particular component added |
| SpecificComponentChanged | Indicates change in given device description |
| SpecificComponentRemoved | Indicates given registry entry removed |

#### 3.2.2.2.3 Protocols

While each component in a station provides interfaces that allow it to connect with other components, components do not decide on their own to initiate connection to other components; rather, the logic for deciding what kinds of components are needed for a given clinical function, for locating and leasing these components, and for telling these leased components how to connect to one another is the responsibility of a station's ***protocols***. While a protocol exists to manage the execution of a clinical function, it accomplishes this by acquiring the services of other components and then configuring them in a way that satisfies the protocol's clinical objectives. Given this, a protocol will possess knowledge of the kinds of components and services that can be used for these purposes. It can also be programmed with knowledge about acceptable "fall back" positions that can be pursued if its standard approach cannot be supported. For example, suppose that a protocol exists to control the collection of images from a high-resolution digital scope of some sort and to transfer the data generated by this scope to the display and storage of some remote station designated by the local operator. The protocol's default operation may be to acquire the scope specified by the operator (if there is more than one installed on the station), a local display component, a local control component, and display and storage components for the designated remote station. In addition, the protocol would need to establish a communications channel to the remote station so that data from the scope could be fed "live" to the remote station for viewing and archiving. Now suppose that, in the process of surveying the available resources, the protocol decides that it cannot establish a communications channel capable of supporting the bandwidth that the scope is capable of producing. Rather than simply giving up and declaring to the operator that the function cannot be executed, the protocol now begins to look for alternative solutions to offer to the local operator. Depending on the nature of the components that populate the local station and the clinical requirements, these might include things like instructing the scope to operate at a lower resolution or frame rate, using local storage capabilities to buffer the difference between the scopes higher data rate and the communication channel's lower rate, or simply storing the data to local record space and then forwarding it to the remote station once the operation is complete.

One important point to note is that, because one of the chief goals of this architectural approach is to allow mix-and-match composition of stations and systems from independently developed components, some of the questions asked today by engineers who integrate components into systems will be asked by the protocols in tomorrow's systems. In addition, some of the information contained on "spec sheets" today that the engineer uses in figuring out if different components can be interconnected will have to be contained tomorrow in the components themselves. In this way a protocol can dynamically inspect the characteristics of components that it may want to lease and

determine which of the components in a system are usable together and under what circumstances.

In as much as protocols are themselves components, they support each of the core interfaces discussed so far. They can register themselves with a station on which they are installed. They drive the subscription of components that they release and can themselves subscribe to messages published by other components (including other protocols). As already described, they interact with registries for the purpose of acquiring resources needed to support the functions that they exist to manage.

As with other core components, a protocol maintains internal data sets that describe the protocol or that serve as internal "scratchpads" for recording information relevant to the protocol's operations and also provides service interfaces that are unique to the operation of the protocol. In particular, a protocol contains a list of options for implementing the protocol's function (i.e., a list of what kinds of components are needed and how they can be stitched together to deliver the functionality), a list of the specific components that the protocol has actually leased, a list of the state variables for which the protocol will publish notifications when changes in the variables occur, a list subscribers to events published by the protocol, a service interface that is standard across all protocols and one that is unique to the individual protocol.

The first data set – the protocol description – parallels the structure of the general component's description with a few differences. Fields in this description include:

| **Protocol Description** | |
|---|---|
| Registry Component ID | Unique ID assigned by registry |
| Registry Location | Address of the registry |
| Component Type | = Class + Type |
| Class | In this case "PROTOCOL" |
| Type | Uniquely identifies component's function |
| Component ID | = Device Manufacturer + Model + Serial # |
| Device Manufacturer | Uniquely identifies the device's manufacturer |
| Model | Unique within the manufacturer's product line |
| Serial Number | Unique within a products of a given model |
| User-assigned Name | Plain-text name assigned by owner/admin |
| Component Location | "Base address" of the device |
| Component Description | Description of protocol for use by other tools |
| Component Persistence | Indicates if should be treated as "permanent" |
| Security-related Parameters | Tree listing protocol's security capabilities |
| Service List | ={Service Type+Service Location+Msg List} |
| Service Type | Specific set of capabilities component offers |
| Service Location | Address of this set of capabilities |
| Message List | ={Msg ID+Direction+Content+Encoding+Prot} |
| Message ID | Unique within scope of component |
| Direction | Indicates if message is sent or received |
| Content | Variable/parameters contained in message |

| | |
|---|---|
| Encoding | Structure of specific fields & entire message |
| Protocol | Indicates protocol used to communicate |
| Message Location | Used to override Service Location, if desired |
| Lease Type | Indicates “exclusive” or “shared” lease |
| Share Count | If “shared”, gives number of shares allowed |

The second data set stored by the protocol is its “Protocol State”, a subset of the state information discussed in the Generic Component.

| **Protocol State** | |
|---|---|
| Registration Status | Indicates whether protocol is registered |
| Subscriber Count | How many subscriptions are established |

Next comes the list of other protocols which have subscribed to the protocol:

| **Subscriber List** | |
|---|---|
| Subscription ID | Unique for all subscriptions & all components |
| Subscriber | = Station ID + Registry Component ID |
| Station ID | Universally unique identifier for station |
| Registry Component ID | Unique ID assigned to subscriber by registry |
| Service Type | Specific set of capabilities of interest |
| Message ID | Specific message of interest within this set |

The fourth data set stored by the protocol is the “Implementation Descriptions List”. Each “Implementation” that populates this list is a specification that describes a list of component types and the list of component interconnections and supporting channels that can be used to satisfy the protocol’s function. In this sense, a “component” in the first list is a role that some real-world component will need to fulfill once leased by the protocol and stitched together with other components by means of subscriptions. Implementations are listed in order of preference with “Implementation 1” being the preferred approach to delivering the protocol’s function and the last implementation being the approach of last resort. Implementations are described by the following fields:

| **Implementation Descriptions List** | |
|---|---|
| Implementation ID | Identifies an allowable protocol implementation |
| Implementation Description | Describes nature of this implementation |
| Component List | Identifies components needed for implementation |
| Protocol Component ID | Identifies a role in the implementation |
| Component Type | Identifies type of component needed in this role |
| Class | Describes the general type of component |
| Type | Describes the unique characteristics of component |
| Channels List | Identifies channels used to support connections |
| Channel ID | Uniquely identifies a given channel role |
| Channel Type | Specifies “synchronous” or “asynchronous” |
| Interconnections List | Maps components, messages, and channels |

| | |
|---|---|
| Connection ID | Identifies logical connection in the implementation |
| Source | Role component from which message flows |
| Sink | Role component to which message flows |
| Service Type | Interface with which message is associated |
| Message ID | Message which is to flow. |
| Channel | Logical channel that will supports the flow |

The next data set – the leased components list – tracks which implementation the protocol has selected, which components have been leased to fill each role in that implementation's component list, and which connections have been successfully established. Fields in this data set include:

| **Leased Components List** | |
|---|---|
| Leased Components | List of which leased components fill which role |
| Protocol Component ID | Identifies the role being filled |
| Station ID | Identifies station of leased component |
| Registry Component ID | Station-unique ID of leased component |
| Connections | List of connections implemented for protocol |
| Connection ID | Identifies connection role being implemented |
| Subscription Status | Identifies whether subscription established |
| Source Attachment Status | Identifies if source has attached to channel |
| Sink Attachment Status | Identifies if sink has attached to channel |

At a minimum, every protocol provides a service interface that responds to two messages:

| **Protocol Control Messages** | |
|---|---|
| StartProtocol | Sent to protocol to start its operation |
| EndProtocol | Sent to protocol to terminate its operation |

A protocol's subscriber list is no different than the list described for the generic component. Protocol event messages to which another component (typically user interface components) might subscribe include:

| **Protocol Event Messages** | |
|---|---|
| ProtocolReady | Sent to subscribers when interconnections done |
| ProtocolInterrupted | Sent to subscribers when protocol stalls internally |
| ProtocolAborted | Sent to subscribers when protocol cannot finish |
| ProtocolTerminated | Sent to subscribers when protocol vacates leases |

Like other components, a protocol will also present to other components a protocol-specific set of messages that allows these components to make use of the services that uniquely define the protocol and, possibly, to dynamically alter some of the static choices embedded in the protocol's "implementations". For example, a pulse oximeter protocol might include a set of commands focused on device control ("start", "stop", "calibrate"), a set of commands related to the user interface ("display/hide SpO2 waveform"), and a

set of commands related to a limited degree of rewiring of subscriptions ("(do not) route average saturation to current patient record").

#### 3.2.2.2.4 Contexts

While it is possible to create stations that offer the same fixed set of services to all station users, more often than not, the ability to customize the station for each user is a desirable thing. For example, a patient visiting the doctor may want to make all of her patient record data available to the doctor and to have any records generated during this encounter added to this same global record. It would be nice to have an easy way to tell the doctor's station where the patient's records are stored. Likewise, a caregiver making rounds in a hospital may want continuous access to his own set of "tools" (analytic software, protocols, etc.) as he moves from room to room. Finding an easy way of allowing this on-the-fly customization of stations to occur would quite beneficial.

To address these needs, the architecture provides for the notion of a ***context***. This "root" component specifies which other components will be employed on a station whenever a given person is interacting with that station. This includes identifying, among other things, which user interface elements will be enabled, which protocols will be used, how these components (i.e., the user interface elements and protocols) are to map to each other to form the station's core, and which repositories are to be used for recording transactions in which the person is involved.

Three kinds of contexts are defined by this architecture. The station's ***default context*** specifies how the station is to be organized under most circumstances. The ***caregiver context*** specifies how a given clinician prefers to operate. When introduced to a station, this context can extend the station's default capabilities by causing new components to be added to the station (note that this may occur by downloading to the local station or by establishing links to remote components) and integrated with existing components or may result in a partial replacement of station components. Introducing a ***patient context*** to a given station provides that station the information it needs to access and interact with as much of the patient's record base as the patient allows and, in certain situations, to create on the fly a station that supports a patient's unique needs.

In a very real sense, a station's default context and its registry constitute the heart of the station. When the registry boots up, the default context, being one of the registry's standing subscribers, is notified of this event. In response, the default context, which functions like a protocol, goes looking for a user interface framework, user interface components, and protocols that it will lease and then knit together through the process of subscription. Once established, these interconnected components form the core of the station and enable the operation of its default capabilities.

When a caregiver context is introduced to the station, the default context is notified of this event. Using the data stored in the caregiver context, the default context revisits the station organization question once again. If new (i.e., unlike any handled in the default context) user interface components or protocols are specified, then these components are leased and then stitched into the station's infrastructure. If given components or protocols are specified for which similar existing elements have already been activated,

then the existing components and protocols are removed from active service and replaced with those preferred by the caregiver. If the caregiver context specifies a repository for recording patient records, then records will be routed to the particular components specified by the context. Note that this does not replace any standing recording to default locations and is only meant as means of allowing a caregiver to retain their own personal copies of records generated by patient encounters.

In like fashion, a patient context allows the patient to carry their own user environment with them both for their own use and for the use of caregivers who are tending to them. For instance, a patient sitting down at a "do it yourself" blood pressure monitor in the grocery store could introduce his context to the machine and have readings automatically forwarded to one or more patient record repositories (e.g., his own personal repository and his doctor's). Similarly, a patient checking into a hotel on business travel might use an intelligent "access point" to connect a suite of medical devices that the traveler owns to a collection of resources (e.g., HAVI-enabled television and Internet access) found in the hotel room. Once attached to the access point, the patient's context would be used to find needed resources and to then connect external resources, such as processing resources, protocols, or patient records that the patient uses (e.g., during exercise).

When a context is removed from a station, the default context on that station once again reviews the station organization. Components and protocols unique to the removed context are disconnected and released and components and protocols that had been released to support customization are once more restored.

#### 3.2.2.2.5 The Internal Communications Bus

As was discussed above, a telemedicine station may be a tightly integrated, monolithic platform or it may be a distributed collection of components that federate as needed to accomplish some clinical task. Whereas the first approach is likely to rely on operating system mechanisms to achieve inter-component communications, the second will rely on some sort of network. It is this second approach that is the subject of this section.

In a distributed, network-based station design, the network itself can be viewed as a compound (i.e., multi-part) component that we will refer to here as the station's ***internal communications bus***. This bus is treated as being divided into two main parts: the media and transceivers used to move bits between components and a set of logical connections or "channels" that are dynamically established for a time between different components within a station.

In practice, the physical elements of the bus may not be a single network but may consist of several different interconnected network segments that are based on different technologies (Figure 47). For example, in a home network, a "home gateway" might connect to the Internet via external cable modem or DSL (phone line) modem and to internal Ethernets running over phone lines to support home computing needs, to internal fiber optic "Firewire" buses that support the home's audiovisual components, and to a power line network that is used to support home automation. In this configuration, even though the home's internal network consists of three different network segments and

three different kinds of media, the home gateway bridges the segments to allow for communication between devices irrespective of location.

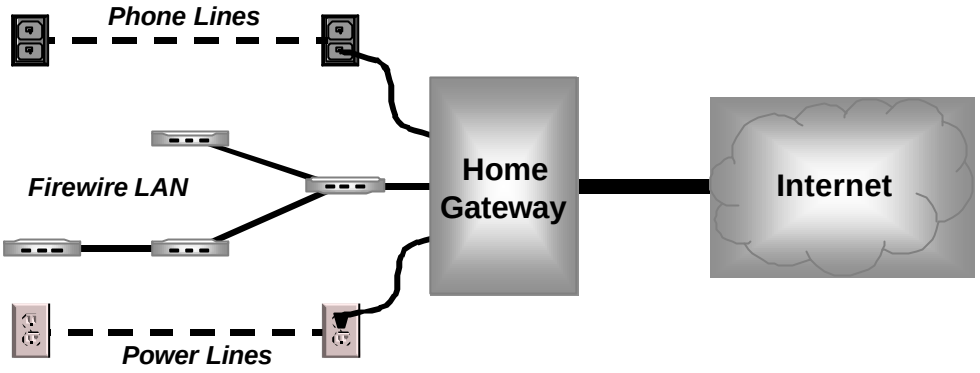


**Figure 47. A Home Network**

Station components connecting to a network segment will do so in one of two ways: directly or by means of a ***network access point*** (Figure 48). In the first case, the device supports the network segment physical layer as well as the networking protocols used to move bits between the device and other network nodes. In the second case, a specialized box is used to translate between the native communication capabilities of one or more devices and the more general communications format of the network segment. One example of this would be a device that allows components that communicate using an infrared link to become nodes on a wired Ethernet network.

On the logical side, the bandwidth available for component-to-component communications is divided between a persistent channel that supports all "administrative" traffic within the system and some number of other channels that are dynamically created

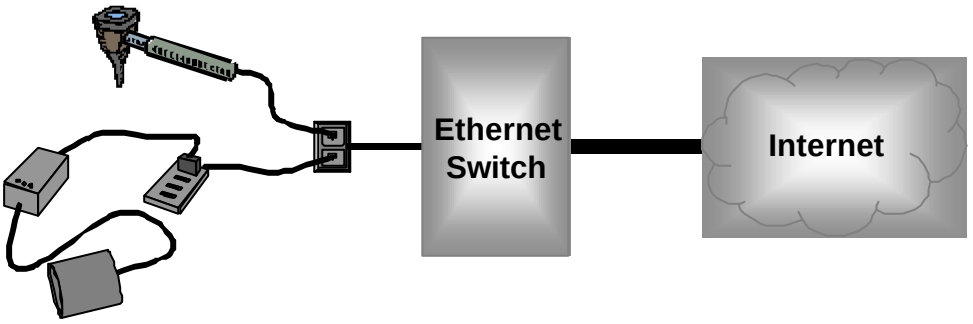


**Figure 48. Two Types of Network Access**

when needed and destroyed when no longer needed. With the exception of the administrative channel, all communications within a station are built on one-way communication links. In this approach, a single data ***source*** passes data to a channel that terminates in one or more data ***sinks***. From the perspective of a component, the logical communication entities with which it deals are as shown in Figure 49. Each channel with which a component must interact is instantiated via a proxy that moves messages onto

and off of the underlying media. As shown in Figure 50, the movement between the channel and a specific physical medium is supervised by a ***media manager*** that exists to hide the differences related to various media that might be used in a station.

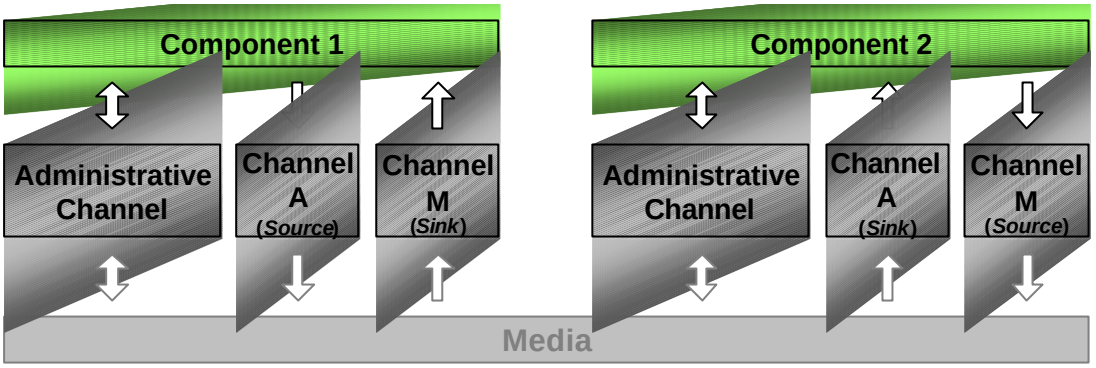


**Figure 49. Component Interfaces to Internal Communications Bus**

The motivation for using channels as described in this section is that communications within a telemedicine station fall into two broad categories: asynchronous and synchronous. Asynchronous communication delivers data in "chunks" on an as-needed basis. Synchronous communication is used in situations where data needs to be continuously streamed from a source to one or more sinks. Examples of the first type include individual commands or status messages sent between components (e.g., "Start" or "Operation completed"). Examples of the second include continuous waveforms (e.g., data from an ECG lead or the audio signal from a stethoscope) and video data (e.g., from a videoconferencing link). Whereas the first can be delivered "eventually" (within reason), the second can levy strict requirements on the performance of the station's

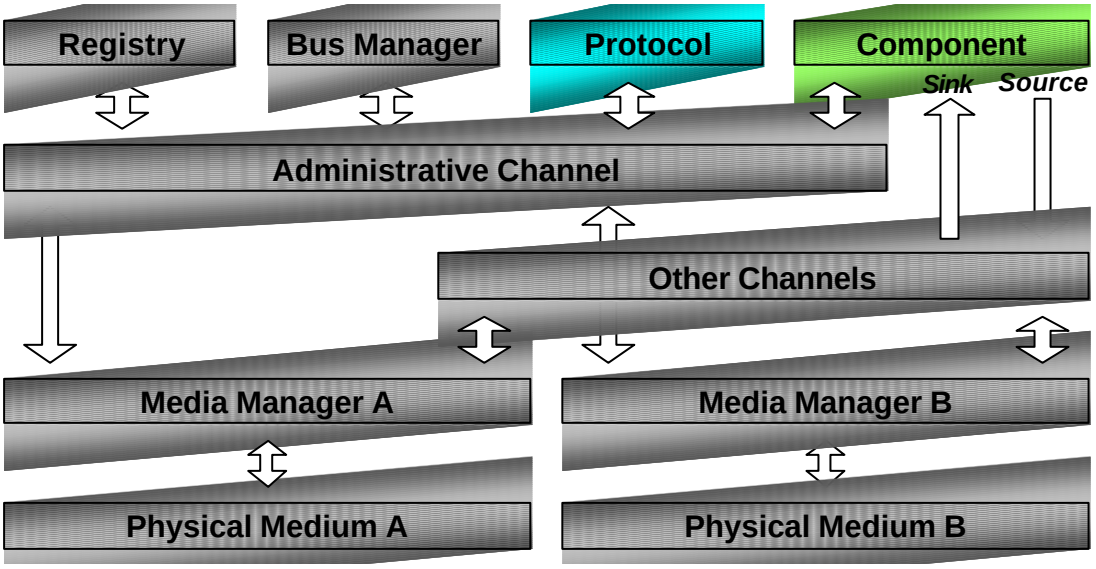


**Figure 50. Internal Communications Bus Architecture**

communication system (e.g., delays and irregular timing of delivery in voice communications can make person-to-person interactions awkward and uncertain performance of a remotely operated scalpel might be fatal). Given this, for synchronous communications, the station's communication bus must be able to provide certain quality of service guarantees. In the internal communications bus architecture, this responsibility falls to the media manager. In addition, depending on how components are distributed, it may necessary to secure certain communication links within a station and between stations. Channels provide a mechanism for delivering fine-grained control over secure communications within a station.

In preparing to use the bus, a protocol will first interact with the station's ***bus manager***, requesting that a channel with specific characteristics be established (e.g., synchronous, of a specified bandwidth, connecting specific components that have been registered with the station). Based on the address information associated with each component, the bus manager identifies which network segments within the station are involved in the protocol's request. Requests to establish a new channel (which is named by the bus manager) are sent to all of the media managers associated with these segments. Each media manager decides if it can honor the request and, if so, reserves the resources needed to create the channel.

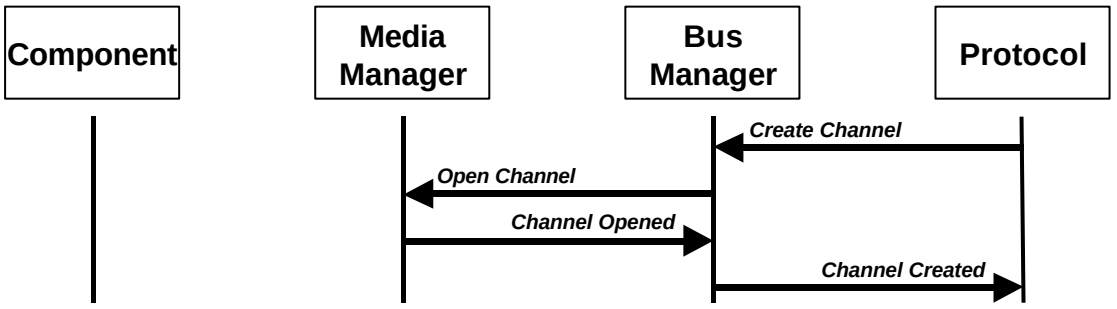


**Figure 51. Protocol Acquiring a Channel**

If the resources needed to support the requested channel are lacking, then the bus manager notifies the protocol of this fact. As with other components, the protocol may request that the bus manager tender to other channels a request for vacation of bandwidth. If adequate offers are returned, then one or more offering channels are closed down or altered (e.g., a channel's bandwidth is reduced) and a new one created for the requesting protocol. If the offers are inadequate, then the protocol is notified of this fact. In response, the protocol may mandate the establishment of the channel (this may or may not be honored) or may simply inform the station user that the resources needed to support the operation are not currently available.

In the event that a protocol's request to create a channel involves one or more "permanent" components that are not currently attached to the station, the bus manager will notify the protocol that the request cannot be honored until the missing components are attached (ideally, a well-formed protocol would have known of this problem and

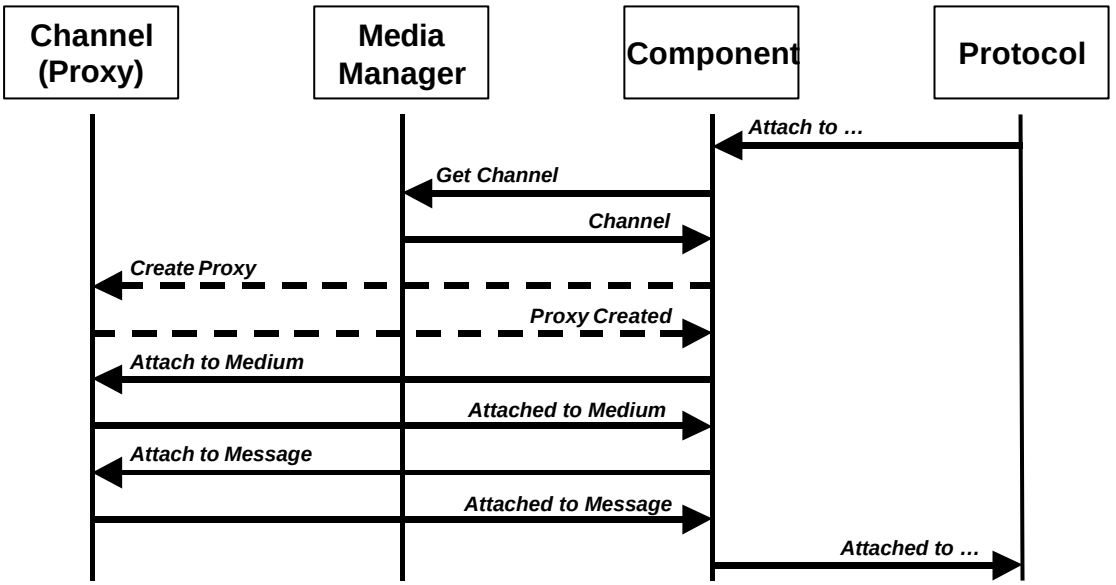


**Figure 52. Attaching a Component to a Channel**

rectified it before ever making the request). If a leased component is removed from the station once a channel has been established and the leasing protocol (informed by the registry of this event) decides not to vacate the lease, then the protocol directs the appropriate media managers to destroy the channel that has been associated with the removed component (in practice, this may be multiple channels). When the component is reattached to the station and its new address is known, the protocol once again works with the bus manager to create a new channel(s) needed to support communications. Finally, if the network segment supporting a channel fails, the associated media managers will notify the bus manager, which, in turn, will alert the affected protocols.

#### 3.2.2.2.6 Device Buses and Factories and Proxies

As discussed above, not all devices will attach directly to the network segments that make up a station's internal communications bus. Even if a station employs a monolithic design, devices may not plug directly into the station itself. In both cases, **device buses** may be used to enable plug-and-play devices to be easily attached to and removed from a station. Common device buses in use today include Universal Serial Bus (USB), IEEE-1394 ("Firewire"), the IrDA (the infrared ports used on many computing devices), and Bluetooth (an RF-based proximity networking standard).

The key feature of a device bus is its ability to dynamically detect the presence of a new component or the absence of a component that had been present. When a device is attached to this sort of bus, there is usually a period of negotiation on the bus in which the bus assigns the new device a bus-unique address and interacts with the device to determine something about the nature of the device that has been added to the bus. Once this is done, the "device drivers" that monitor the bus on the network access point (or on the computing platform) announce the presence of a new device to the rest of the system

to which the bus is attached. This allows the system to then launch whatever software is needed to interact with the newly installed device.

While one might hope that an interoperability architecture like that being described in this document would eventually lead to the development of devices that are fully compliant with the architecture's specification, the reality is that full compliance for all devices will probably never be achieved. There will always be devices that employ their own proprietary interfaces. Even so, if dynamic composition of a telemedicine station from plug-and-play components is ever to become a reality, devices of a given type must present a standardized interface to other components within a system. To handle this disparity, the architecture introduces the notion of a ***device proxy*** – a piece of software that on its station-internal side presents a standardized interface to the rest of the station's components and that on its device side presents an interface capable of handling device-specific interactions. In between, this body of software translates between the two interfaces. The effect is to "wrap" non-standard devices in ways that make them appear to the rest of the station to be standard-compliant devices.

As shown in Figure 53, devices will come in a range of capabilities. In an ideal world, a device that attaches to a station would be able to plug directly into a station's internal communications bus, would support all of the standardized service interfaces defined for that type of device, would be able to uniquely identify itself, and could describe itself to other devices when queried. By comparison, "intelligent devices" might do all of these things with the exception of being able to plug directly into the station's internal communications bus. For these devices, connection would be by means of a device bus and a "light proxy" that could route the device's communications between the device bus and the internal communications bus. At the next step down the ladder of device intelligence, the "self-aware device" is able to publish its unique device identifier but knows nothing of its own capabilities. To make up for this deficit, its proxy embodies all of the self-descriptive capabilities needed by a device wanting to register with a station. The proxy may also translate between device-specific services and the standardized service interfaces established for that particular kind of device. At the bottom step of the

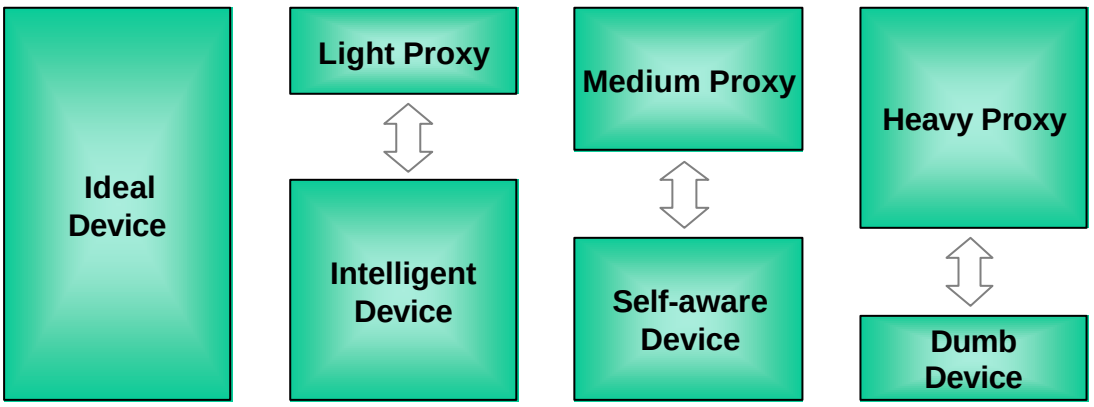


**Figure 53. Devices and Proxies**

ladder is the "dumb device". This component knows nothing about itself including its identity. All that the device can do is to accept and respond to commands – most likely using its own proprietary protocol. It is simply an engine that provides some clinical functionality. All of the higher-level intelligence that makes it look like an architecture-compliant device is vested in its "heavy proxy". While all four types of devices might be used in the kinds of telemedicine stations that we are trying to produce using this architecture, the first three types of devices (because of their ability to self-identify, as discussed below) are better suited for use in this environment and, of these, the first two are preferred.

When taken together, the device bus and proxy mechanisms work as shown in Figure 54. Attaching a device to a device bus causes a bus event that leads the drivers that control the bus to determine what has just been attached to the bus and to reconfigure the bus as required. The drivers then notify a factory component of the new device's existence on the bus at a given bus address. The factory then creates a proxy component that begins to explore the device (or some other information source) to harvest the device details needed by the proxy. Once these details have been collected, the proxy begins the process of registering the new "device" (which is a combination of the physical device and the proxy) on the station.

### 3.2.3 Notes on Specific Component Types

While the structures and interactions that have been discussed so far constitute the core of the architecture, they are still nothing more than infrastructure. The real clinical

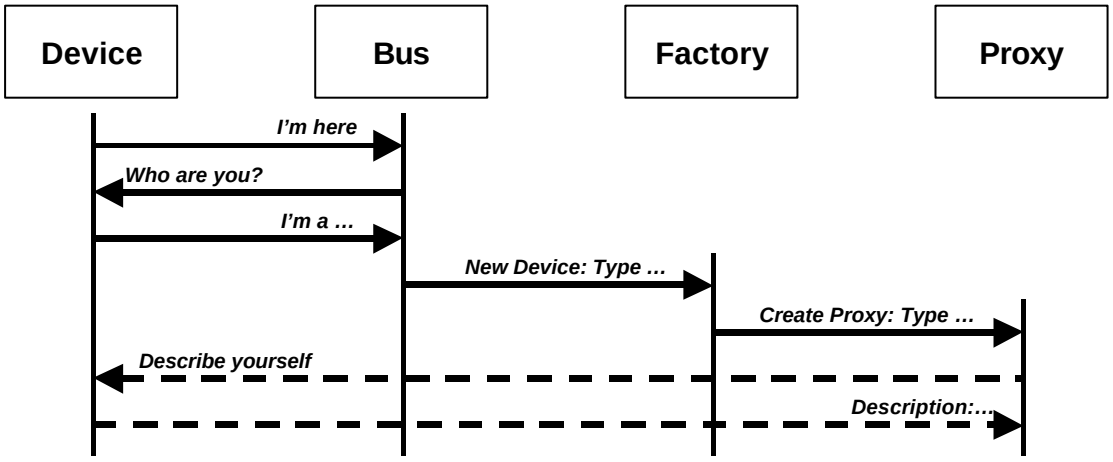


**Figure 54. Adding a Device to A Bus**

functionality delivered by a telemedicine system is vested in four kinds of components: medical devices, patient records, user interface components, and processing components. For this reason, the architecture specification must also include application-level interface specifications for each of the service interfaces provided by individual component types belonging to each of these component classes.

For any given service interface, there will be at least two different component types belonging to different component classes that will implement the interface. For example, in Figure 55, the "Pulse Oximeter Control Interface" defines a set of messages that relate to controlling a pulse oximeter and monitoring its operation. These messages are embodied in both the pulse oximeter component and the pulse oximeter user interface component. For example, the user interface component might know how to generate a "start" message while the pulse oximeter component knows how to process that message. Also, as is implied by the "Waveform Data Interface", "implementing" an interface means that the component knows how to generate or process a given set of messages; however, it does not meant that each component implementing a given interfaces necessarily handles the interface's messages in a uniform fashion. In the case of the user interface component, receiving a waveform data message results in the waveform being displayed. In the statistics component, it results in the waveform being analyzed and key features being extracted. In the patient record component, receipt of the message results in the data from the message being stored to the "current record".

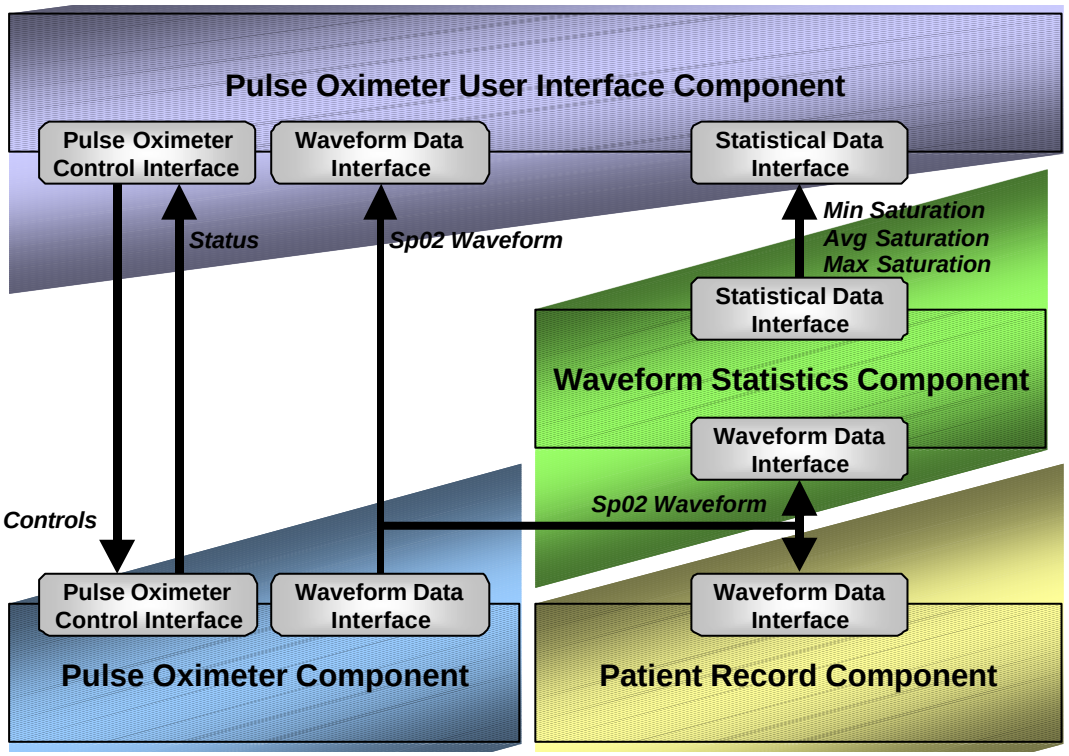


**Figure 55. Standardized Service Interfaces**

### *3.2.3.1 Medical Devices*

In reading through the work that has been done on medical device standardization, it becomes clear that the general device pattern above will need to be extended to include service interfaces for device management (to include inventory management and calibration), device configuration, device control, and data acquisition (Figure 56).

A survey of the telemedicine field shows several classes of medical devices currently in common use today. These include:

- Various kinds of scopes and cameras
- Certain types of radiographic devices
- Stethoscopes
- Vital signs devices
- Point of care test devices
- Other diagnostic devices
- Certain therapeutic devices

In addition, as telemedicine matures and we are able to do more things at a distance, we are likely to see telemedicine-based remote monitoring move from simple monitoring of the patient to monitoring of both the patient and the patient's environment. If a doctor deals with allergy sufferers, it is not hard to imagine a future in which the doctor is able to remotely assess the types and levels of allergens present in his patients' homes.

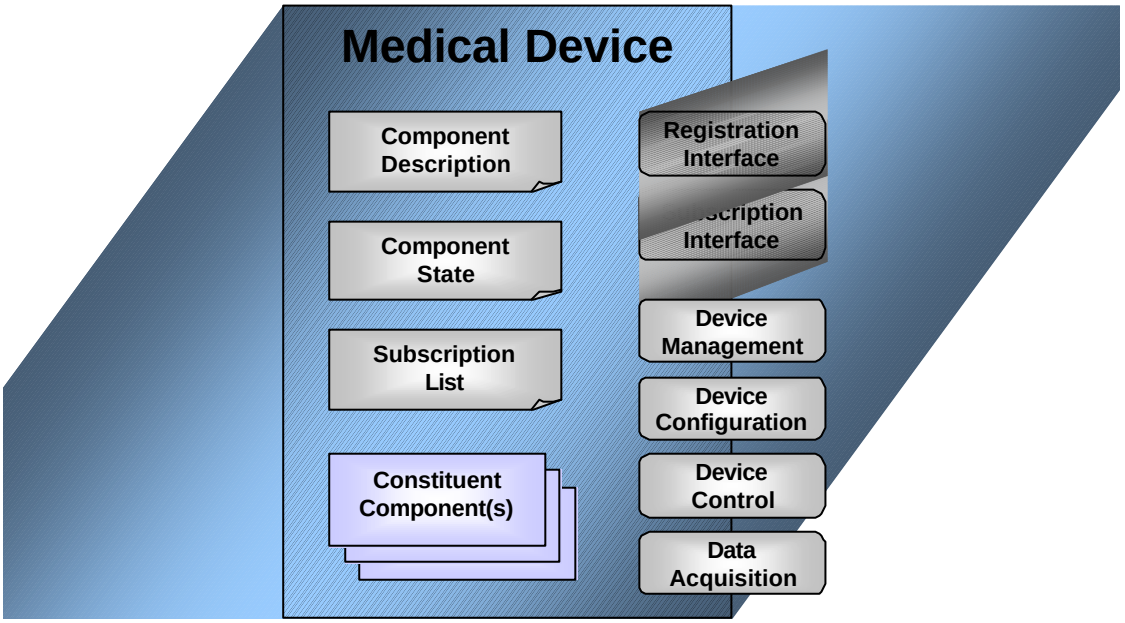


**Figure 56. Structure of the Medical Device Component**

All of these devices can be grouped into categories based on the kinds of data that they produce, as shown in the following table:

| | |
|---|---|
| Still Imagery Devices | ***Digital camera, portable x-ray unit*** |
| Moving Imagery Devices | ***Video camera, otoscope, ultrasound unit*** |
| Audio Device | ***Stethoscope*** |
| Waveform Devices | ***ECG, pulse oximeter*** |
| Scalar Data Devices | ***Glucose meter, blood pressure monitor, weight scale*** |

Given the diversity of data types represented, it is clear that a one-size-fits-all solution probably won't satisfy all needs. In addition, in as much as certain medical device

standards already have significant constituencies, it is likely to be incumbent on the telemedicine community to make our architecture fit the existing standards and not the other way around. For these reasons, it is likely that telemedicine station architectural patterns applied to the medical devices will look something like Figure 57.

Device buses 1 and 2 represent different bus technologies (e.g., Firewire and BlueTooth). Each bus has its own manager that hides the details of device location and bus organization from the rest of the station. When a device is added to the bus, the associated factory creates a matching device proxy on the station. If the device is only "self-aware" (as described earlier), then the proxy then queries the device that it represents and fills its own internal structures (per Figure 56) with the data harvested from the device. If it is "dumb", then the proxy may end up filling its internal structures from a file stored on the station using data entered during an earlier device configuration session. In the case of intelligent devices, the proxy initialization may entail no more than establishing the correct mapping of station-internal addressing to device bus addressing.

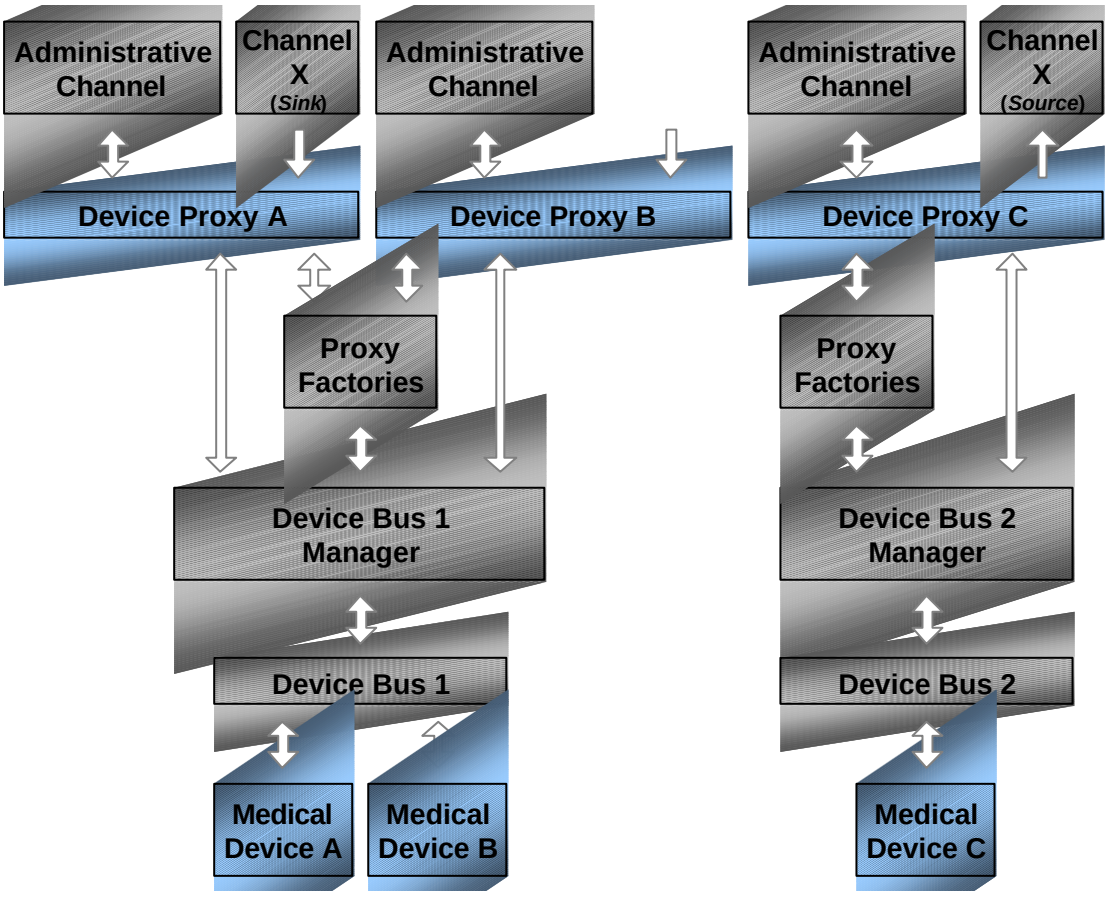


**Figure 57. Medical Device Portion of Station Architecture**

In the process of establishing this addressing, either of two scenarios is possible. In the first, the device that is hosting these proxies already has its own address that it somehow

extends to provide each proxy its own unique network location (e.g., the host device might have a fixed IP address and dynamically assign ports to each of the proxies that is launched in its computing space). Alternatively, each of the proxies might request and receive its own address at the time of its creation (e.g., through the use of Dynamic Host Configuration Protocol).

Once a proxy has been fully configured, it announces the device's presence to the station. This begins the registration process described earlier in the chapter.

### 3.2.3.2 Patient Record Repositories

Because of the wide variety of employment scenarios that telemedicine-based care delivery enables, telemedicine systems must be able to implement a range of patient record storage and retrieval schemes. As noted earlier, these include local storage of records (e.g., in a patient station in order to allow readings to be queued up and periodically harvested by a caregiver station), remote storage (e.g., as in the case of a caregiver visiting patients in a hospital and wanting to have all encounters recorded in the record server in his office), and distributed storage (where all records regarding an individual, even though maintained by different organizations, appear as a single coherent record to a station querying the records). The patient record repository structure in this architecture must handle both the writing and reading of records. Also, because of the desire to handle a variety of media (disk-based records, patient card records, etc.), the architecture needs to address both the logical structure of the patient record system and the media on which this system resides.

The architecture's patient record repository structure must support a number of requirements. First, it must allow for the possibility of records related to multiple patients being stored in the same repository. Next, it must allow for multiple ways of indexing the data stored in the record. These should include both encounter-oriented indexing (i.e., these readings and this information were collected during a given session) and topic-oriented indexing (i.e., here are all of the readings of a given type collected for this patient and this is when they were collected). For each kind of reading collected by the patient record, the patient record interface implemented for this purpose should look no different than that of any other component that could make use of the same reading (e.g., the interface on a user interface component or on a processing component). Also, as new kinds of medical devices are added to a station, it should be easy to automatically add the associated record elements to the patient record repository structure. Finally, the architecture must also be able to handle multiple repositories on one station (as in when a patient card containing a patient record is attached to a station that maintains its own repository on disk).

Given these requirements, the record repository structure depicted in Figure 58 is proposed. The core structure is shown in the narrow columns on the diagram. In this structure, the ***repository*** is a single physical container that holds records for some number of patients. Patients who have records in this repository are listed in the ***patient index*** – a structure that points to the location of the various ***patient records*** stored in the repository. A patient record is a container for all information in the repository related to a given

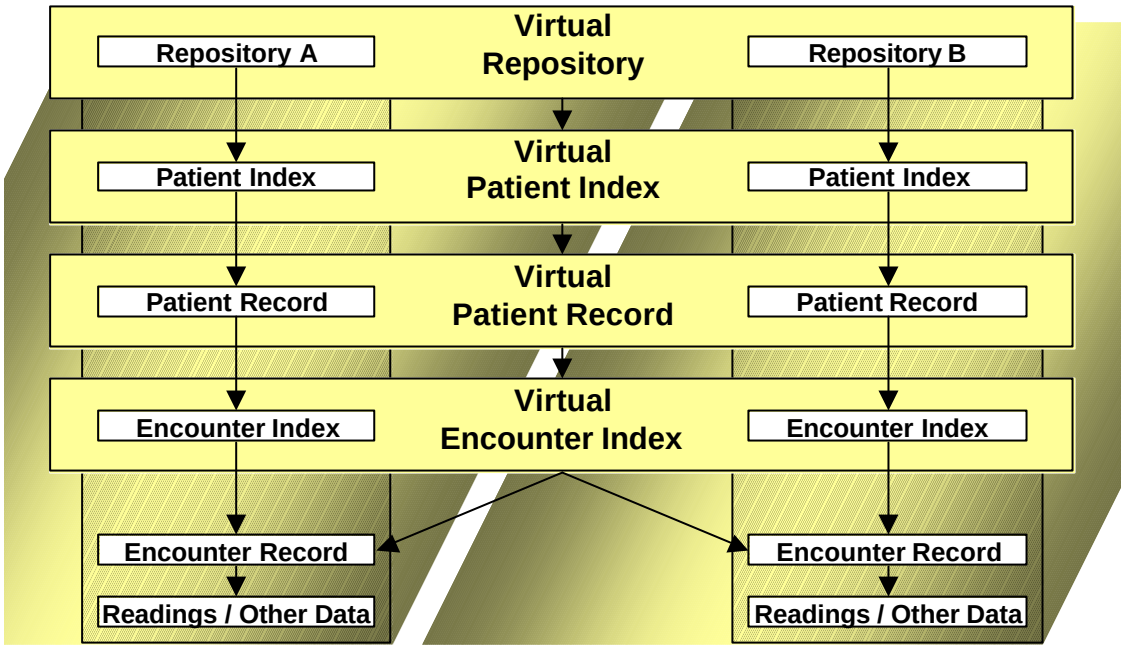


**Figure 58. Notional Organization of the Patient Record Repository**

patient. The information in a patient record is further divided into encounters and the encounters are all cataloged in an ***encounter index***. The encounter index provides information on the location of each ***encounter record*** in the repository associated with a given patient. In turn, the encounter record will contain one or more ***readings*** (e.g., blood pressure, ECG trace) and other kinds of data (e.g., voice recordings or textual data), as appropriate.

To handle the case where a station hosts multiple repositories or the station provides a unified interface to multiple distributed repositories, a ***virtual repository*** structure may be used as the top-level container for all patient data managed by the station (alternatively, each repository in this space can also be presented independently to a station user with a view to allowing the user to explicitly specify which particular respository(-ies) and subordinates structures are to be used in the current activity). Drawing on the patient indexes for each of the constituent repositories, the virtual repository creates a unified ***virtual patient index.*** Similarly, the various patient records for a given patient that are spread across the different repositories are now treated as a single ***virtual patient record*** and all of the various encounters contained in these records are now organized using a common ***virtual encounter index***. Once selected from this index, encounter records and the data they store are accessed directly in the physical repositories in which they are stored.

Accordingly, the patient record component (Figure 59) provides three service interfaces to external components. The media management interface is used to handle issues associated with the media on which the records are stored. Issues such as available storage space, the size of a record, and clearing records from the repository are handled

through this interface. The record navigation interface allows client components (e.g., a record browser user interface component) to navigate through the various repositories to which the station has access and to establish one or more records as the "focus" of subsequent operations that are to be conducted using the record manipulation interface. The record manipulation interface allows other components to create, delete, write to, and read from various structures in the repository. In the case of creating a structure, previous record navigation operations would have set the focus of operations on the next higher-level entity in the repository structure (e.g., creating a new encounter would require first selecting a specific patient's records as the current focus).

Repositories, like medical devices, can possess different levels of sophistication. Some repositories that are accessed by a station will be nothing more than structured sets of data and will lack the "intelligence" to deliver the services of a typical component (e.g., this will be true with many patient card-based repositories). Others will be quite sophisticated, delivering all of the capabilities defined for patient record repository components. Whereas the latter can function as regular components within a telemedicine station, the former cannot. In these cases, a mechanism like that used to create proxies for some medical devices will be used to create a patient record repository proxy that, using the data sets, will provide the services of a regular physical repository. Finally, in as much as some patient record repositories will be introduced to a station via removable media (e.g., patient record cards or a repository on a PDA), the station needs the ability to detect when a repository has been added to or removed from the station.

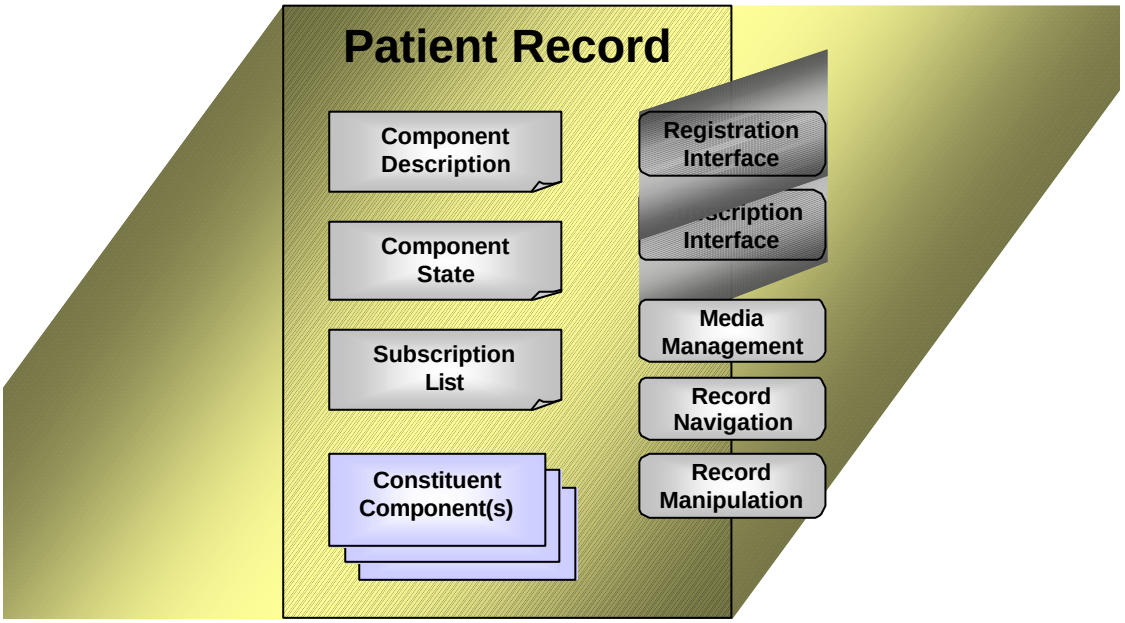


**Figure 59. Structure of the Patient Record Component**

Given these things, it is likely that telemedicine station architectural patterns applied to the patient record repositories will look something like Figure 60. Repository A

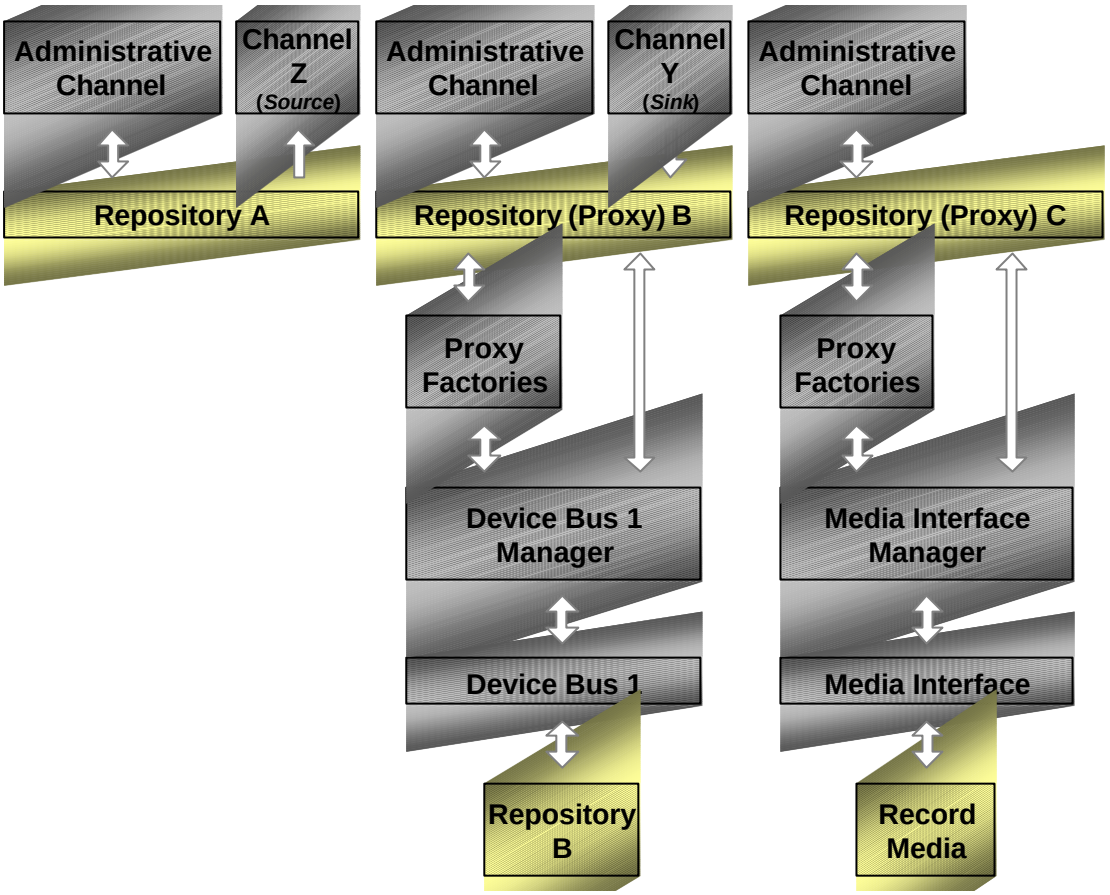


**Figure 60. Patient Record Repository Portion of Station Architecture**

represents the patient record analog of the "ideal device" discussed in the previous section. It is fully functional as a repository and attaches directly to the station's internal communications bus. Repository B is also fully functional but attaches to the station by means of a device bus. In as much as bus-level addressing consideration must be addressed, a proxy located on the station side of the bus routes all communications between the repository and its client components (note that a similar arrangement that vested the proxy with more intelligence can be used to accommodate a repository that does not natively support the standard telemedicine station patient record interfaces). Repository C represents the situation in which the underlying patient record system is nothing more than data stored on some sort of removable media (e.g., a smart card or touch memory). In this case the device is attached to the station by means of some sort of reader. As with device buses, this reader is monitored by a manager component. When the media is detected, the appropriate factory is notified, a proxy is created, and the data from the media is harvested to populate the repository. From this point on, all repository operations, with the exception of writing and deleting, are carried on exclusively with the proxy. Even in the case of writing and deleting, the proxy may control when these operations actually occur (e.g., certain kinds of media may indicate that buffered operations and bulk writes are the preferred method of interacting with the physical media).

Finally, it is important to note that a tremendous amount of effort has been invested for a number of years now in the area of standards for electronic patient records (EPRs) and the results of this work are quite substantial. In addition, the larger patient record market is populated by a number of vendors, several of them quite large. Given this, the intent in proposing the structure described above is not to define yet another standard for EPRs but to provide a reference model for the storage and retrieval of data within the context of an individual telemedicine station or network of stations. While this model, in whatever its final form will be, might become the "native" storage format used in telemedicine systems, given the criticism that telemedicine systems generally do not integrate with other hospital information systems, it is vitally important that bridges be built between this structure and other EPR structures and protocols in common use. This is discussed in more detail below in the section on patient record communications.

### 3.2.3.3 User Interface Devices

There are a number of goals for the user interface aspects of a station's design. As with other devices, it should be possible to add and remove interface resources in plug-and-play fashion. To the extent that different physical user interface devices support it, it should be possible to freely switch which devices are used for controlling and monitoring a given function and for reviewing data generated or stored within the station. It should also be possible to use multiple interface elements for a given purpose (e.g., it should be possible to start a given protocol both by selecting an element on a screen and by using a voice command). Interface controls will come in a variety of forms. They may be integral to the device that they are control, such as a physical start and stop button on a blood pressure monitor, or they may be separate from the device, such as an interface on PDA that is used to control a range of wireless medical devices.

Figure 61 illustrates several of the elements found in the user interface portion of a telemedicine station. At the bottom of the figure is the device that is being controlled and

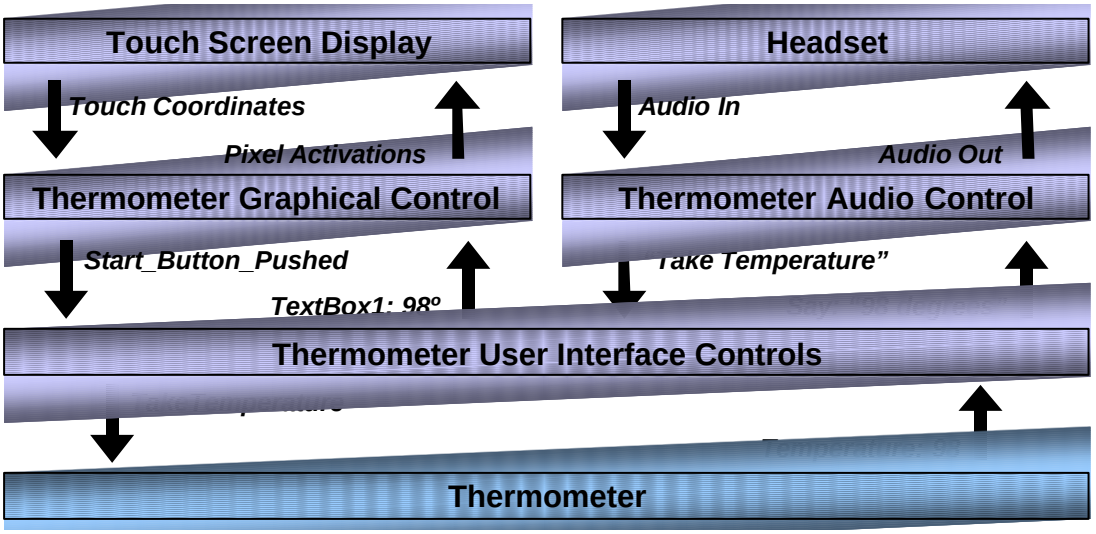


**Figure 61. Relationship Between Components, UI Devices, and UI Controls**

monitored using the interface elements shown. At the top are the physical devices with which the user interacts. These devices generate and accept a limited set of signals, the exact interpretation of which is dependent on their immediate context. In this case, the devices used are a touch screen system capable of graphic display and a headset containing headphones and a microphone. In the layer below this, a graphical control for the thermometer embodies the various buttons and display elements needed to interact visually with the thermometer. Working in concert with the operating system on which this software runs, the graphical control translates the touch coordinates from the display into events that relate to the elements in the control (in this case, they represent the "start" button in the control being selected). In similar fashion, the spoken audio signals picked up by the headset's microphone are decoded as the verbal command, "Take Temperature". Both of these events are translated by the thermometer user interface control into the same command (TakeTemperature) that is sent to the thermometer. The result of this command is data output from the thermometer that is passed back to the thermometer user interface control where it is translated into commands that each of the respective controls can handle. In the case of the graphical control, the instruction is given to update TextBox1 to display the character string "98°". Similarly, the audio control is told to speak the phrase "98 degrees". The result is that the user both sees the temperature reading on the display and hears it spoken through the headphones.

In order to accomplish the goals set for the user interface, both levels of control need to be addressed. While the approach shown in Figure 61, which centralizes all "event handling" related to a component is fairly common, it also makes the goal of creating plug-and-play user interface components harder to realize. As an alternative, the lower control layer (the thermometer user interface control) could be split into two parts corresponding to each of the device-specific controls. Then these controls could be treated as components and readily added to and deleted from a station. While the two-layered organization delivers some flexibility to the user interface design (e.g. "Say: '98 degrees'" could easily be replaced with "Say: 'Temperature normal'" through a change in a configuration file), treating layer pairs as a single component means that developers are free to render the interface components as monolithic entities.

In addition to user interface controls, three other components are central to the user interface portion of the telemedicine station architecture. These are ***contexts***, ***user interface framework***s, and ***protocols*** (Figure 62). The user interface framework is a specialized component that provides access to the "top-level" protocols and/or top-level controls that an operator uses to interact with the system. For example, from the framework, a user might invoke a specific protocol or cause a specific control to be displayed (that, in turn, would invoke particular protocols). As described earlier, invoking a context causes this aggregation of user interface components to be assembled in a way that is tailored to the needs of the context's owner. As with other components, the assembling is achieved by directing all of the components involved to subscribe to one another's services in a particular way (note that in the figure, the channels that would be used to support these subscriptions are not shown).

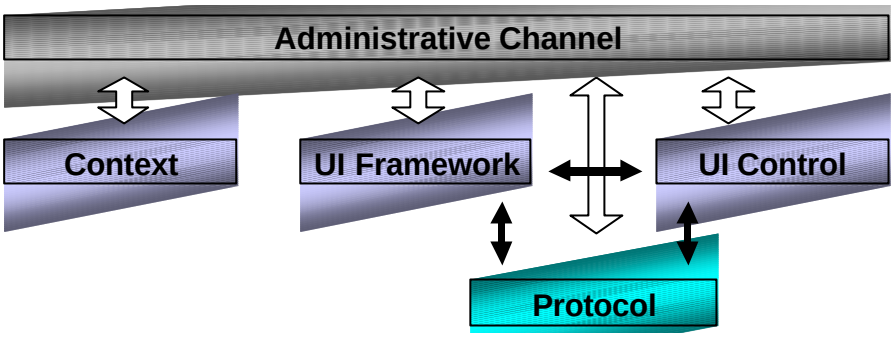


**Figure 62. User Interface Architecture**

In considering how to support plug-and-play on the user interface, three approaches seem attractive. In the first, an external device (e.g., a pocket computer) runs one or more interface applications. Once attached to a station (by wired or wireless connection), the component registration process reveals the presence of the user interface components. The user's context then binds the station-hosted protocols to the appropriate interface elements on the handheld and the user can select and control station operations from this

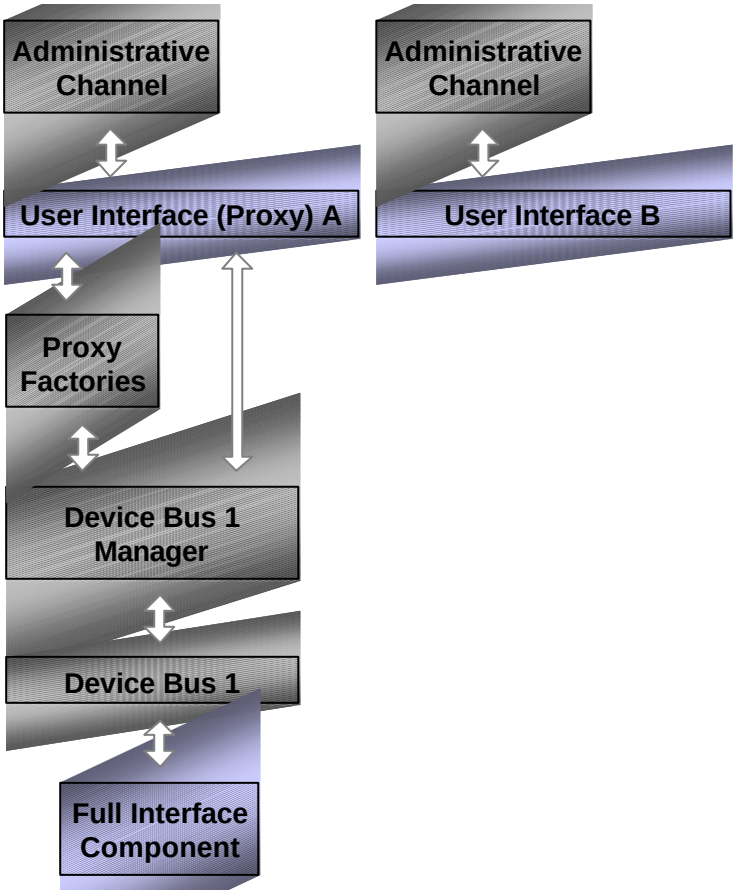


**Figure 63. User Interface Portion of Station Architecture**

device. In the second approach, an user interface framework that is native to the station is capable of accepting new user interface "plug-ins" that extend the framework's capabilities. As needed, new user interface components can be loaded onto the station, installed in the framework, and then, as required by the governing context, mapped to specific protocols. In the third approach, user interface components are downloaded to the station at the moment of need, used for a period of time, and then released (rather than being permanently installed). This last approach is seen in many web-based applications.

Figure 63 depicts the two patterns implicit in these three approaches. In the case of the PDA, the device hosting the user interface components would access the station through a device bus. Because of the addressing issues entailed, a proxy to this device would be spawned on the station to route messages between the user interface components and other components on the station. Note that this approach might also be used to implement a standard user interface for compound devices that possess only proprietary controls. In the latter two cases, the interface elements would be installed as local components (even if for just a short period of time) and would interact with other components in the standard way.

#### *3.2.3.4 Processing*

In most cases, processing elements will be software entities that can register with a station and present themselves to the rest of the station as standard components. At the same time, it is also possible to render a processing capability as a specialized piece of hardware. For example, it is easy to imagine a signal-processing unit that plugs into the station's internal network or that attaches to the station via a device bus and that delivers data transformation capabilities for specific clinical applications, such as the detection of certain types of artifacts in a given kind of data set. As with other devices, these hardware-based components might communicate using standard compliant interfaces or proprietary interfaces. As with the other components discussed so far, the approach taken by the architecture is to proxy the device when required for message routing and when required for the purpose of translating between standard and proprietary protocols.

### 3.2.4 Station-to-Station Operations

Given the station architecture described above, the next question to address is how to design telemedicine systems to allow stations that have never before interacted with one another to discover each other and to begin to make use of each other's capabilities.

#### *3.2.4.1 Communication-Related Components*

As with station-internal operations, station-to-station operations are built on a few key building blocks that are elaborated in different ways to handle the range of station-external communication needs found in telemedicine. These building blocks are the station's ***communication devices***, the ***communication managers*** associated with each of these devices, the ***channels*** that route data between internal and external components and that transform these data as required, an **external communications manager** that provides the station's "dial tone" to external stations, and ***session managers*** that moderate all requests for service by external stations.

#### 3.2.4.1.1 Communication Devices

The foundation of this section of the station's architecture consists of the devices that are used by the station to move bits into and out of the station. These may be standard modems, wired Ethernet transceivers, radios, or any number of other devices capable of supporting point-to-point or networked connectivity. Like many of the devices discussed so far, these components can typically be controlled by a proprietary protocol or one that is standard within their industry (such as the AT command set used in modems). One of the key concerns in this area is hiding from station-internal components the fact that different classes of devices will function in different ways. For example, communicating using a standard modem involves a phase in which the desired circuit is established and, after use, one in which it is torn down. With other technologies, such as cable modems, the connection to the outside world is persistent.

Most telemedicine stations in use today provide a single communications device for establishing external communications. In as much as most current station designs are monolithic, the device is tightly integrated into the station. With the component-based approach advocated in this document, this need not be the case. If desired, it should be possible for a station to employ multiple devices simultaneously and to even share devices across multiple stations. While this might eventually include allowing multiple stations to simultaneous employ the communication resources of a given external communications device, the current thinking here is that a given external device might be used by any station in a cluster of stations but only by one station in the cluster at a time.

#### 3.2.4.1.2 Communications Manager

In order to shield station components from the details of what kinds of devices exist to support external communications and how these devices are controlled, the architecture includes the ***communications manager***. On its station side, this component provides a standard set of services that are used by other components (primarily protocols) to access and control communication resources. On its device side, each communication manager provides the interfaces needed to control the operation of the device for which it is responsible. The job of each communications manager is to translate standardized station-side transactions into device-specific operations and to translate device-specific events into standardized state event messages usable by other components in a station.

As with the media managers on the station's internal communications bus, one of the services provided by the communication managers is the establishment of channels that can be used for external communications. In addition to negotiating for bandwidth to be allocated to a given communications link, this process of establishing a channel involves specifying the protocols to be used on the communications link, the quality of service required, and the security mechanisms to be used during communications.

#### 3.2.4.1.3 Channels

As in station-internal communications, ***external communication channels*** are instantiated as station components that route messages and data streams between the underlying communications media (in this case, the data interfaces of the communication

devices) and other components within the station. As shown in Figure 64, this typically involves the use of an internal communications channel to make the connection.

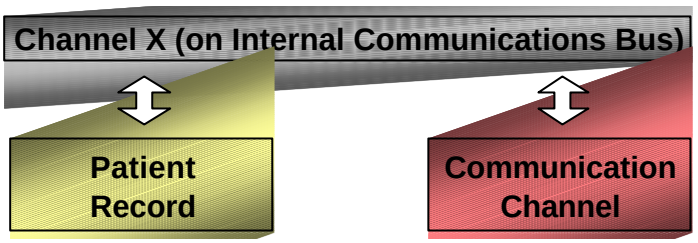


**Figure 64. Interactions Between Channel Types**

Whereas station internal communications are meant to be format-neutral, differentiating only between those messages that levy some sort of quality of service requirement and those that do not, the same is not true of external communications. The intent in the design of the external channels is to allow them to also be able to perform format and protocol conversions as needed to support interoperation with external systems.

#### 3.2.4.1.4 External Communications Manager

Just as the internal communication bus maintains bus manager that governs top-level bus operations, stations that support station-to-station communications will employ an ***external communications manager***. This component exists to control whether or not a session will be established between the station to which this component belongs and one or more remote stations. When other stations come knocking, it is the external communications manager that answers the door. If a session is to be established, then it is the external communications manager that causes a ***session manager*** to be created to govern communications during this session.

#### 3.2.4.1.5 Session Manager

During a session with a remote station, a station's session manager proxies leasing and subscription services offered on the remote station. In the case that multiple stations wish to access a given station at the same time, the station launches a session manager for each of the querying stations and these managers collaborate throughout the sessions to balance needs of all of the client stations. When remote resources are leased, the session manager is responsible for ensuring that proxies needed for these resources are instantiated on both its own station and on the remote station. In establishing secured operations, the session manager serves as the focal point for session negotiation and access controls. Likewise, when quality of service must be maintained with the network, it is the session manager that negotiates and secures QOS agreements from the underlying network and that monitors status reports from this network throughout the session, renegotiating with both the network and the station-internal components as needed to ensure continuity of operation. In the case that more bandwidth is required than the station can support, the session manager works with the components involved and other session managers on the station to negotiate vacation of bandwidth or, at the least, new quality of service regimes.

### *3.2.4.2 Operation of Station External Communications*

In supporting external communications, a station must be able to handle variations in both the kind of connection being made to a remote station and in origin of the connections. Specifically, the station must be able to:

- initiate a connection (when persistent connections are not used or usable),
- accept an invitation to initiate a connection,
- initiate a session with a remote station, and
- accept an invitation from a remote station to participate in a session .

In addition, as a station begins the process of subscribing to remote services or supporting services leased by a remote station, it must be able to instantiate channels that encapsulate specific station-to-station protocols and that translate between these external communication mechanisms and those used internal to the station and to tear these channels down when they are no longer needed.

#### 3.2.4.2.1 Initiating a Connection and Starting a Session

Connection to a remote station is initiated by a station-internal component (typically a protocol) passing the remote station's ID to the station's external communications manager and asking the external communications manager to initiate a session with the remote station. In response, the external communications manager directs the appropriate factory to create a new session manager component. Once this has been created, the external communication manager directs the session manager to initiate a session with the specified remote station. The session manger then checks the station's address book to

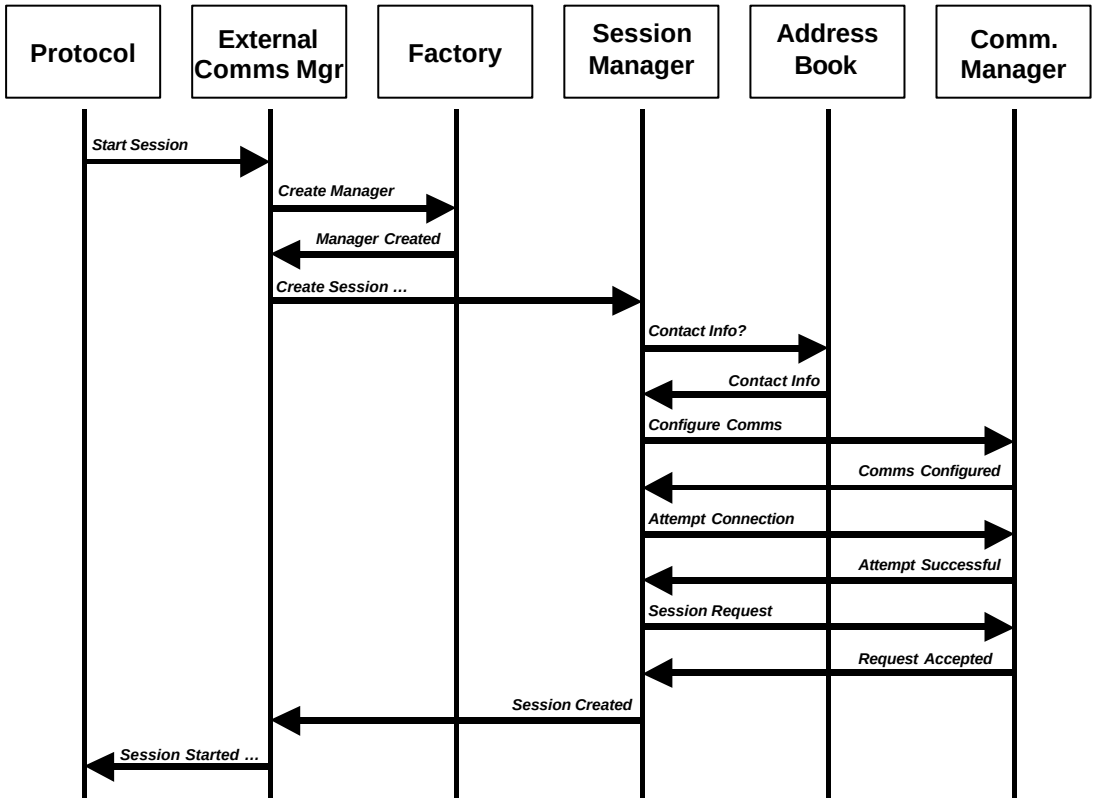


**Figure 65. Protocol Directing Establishment of a Session (with Connection)**

determine the kinds of connections supported by the remote station. In the case that persistent connections are not available on the remote station, the session manager retrieves all of the relevant communication parameters (e.g., phone number or radio ID, encoding schemes) from the station's address book and instructs the communication manager for the requisite communications device to configure its device. After configuration is completed, the session manager directs the communications manager to attempt to establish a connection with the remote station. If the remote connection is successfully established, the communications manager notifies the session manager, which, in turn, notifies the external communications manager. Finally, the external communication manager answers the original request to start a new session by returning to the requesting component a handle to the session manager created to run this session.

For stations employing persistent communications, the interaction diagram degenerates to the following figure.

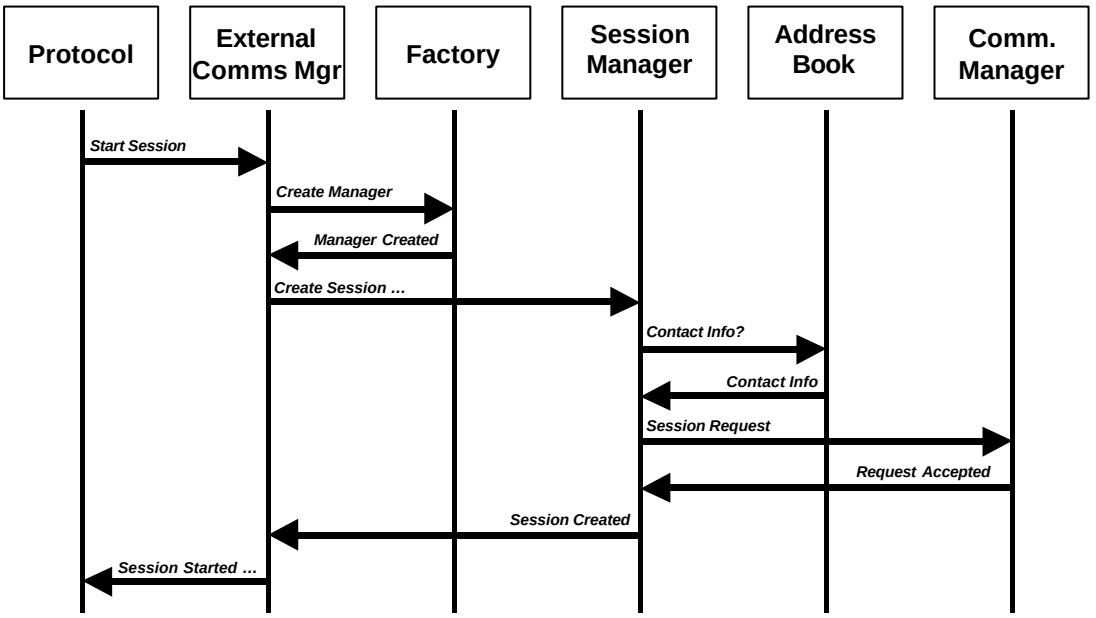


**Figure 66. Protocol Directing Establishment of a Session (with Persistent Comms)**

#### 3.2.4.2.2 Accepting An Invitation To Connect and Start a Session

While all of this is happening on the initiating end of the session, the remote station is responding to the request to participate in the session. The first activity seen by the remote station is a "ring" received by the remote station's communications device selected by the initiating station. The communications device automatically answers the incoming call and negotiates with the initiating station the details of how low-level communications will be carried out. Once this is settled and the stations are able to exchange messages, the initiating station transmits a request to establish a session, which is received by the remote station's associated communication manager. This request is forwarded to the station's external communications manager, which then directs a factory to instantiate a session manager component to manage all subsequent interactions. When this has been completed, the external communications manager returns, via the

communications manager, a message to the initiating station indicating the remote station's station acceptance of the request to establish a session.

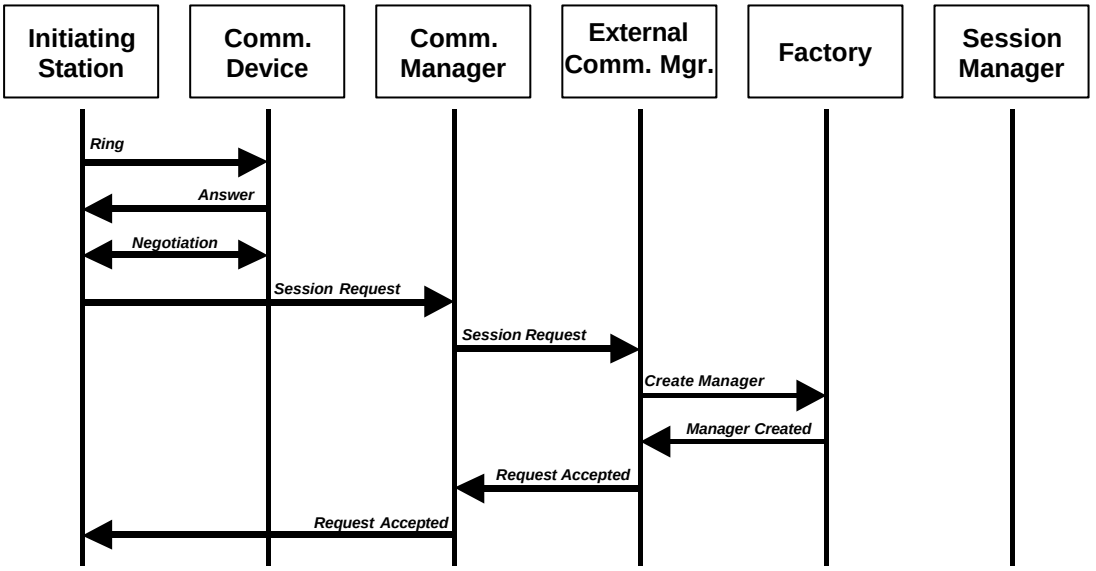


**Figure 67. Station Accepting Request to Establish Session**

When persistent connections are used on the remote station, the interaction diagram degenerates to the following.

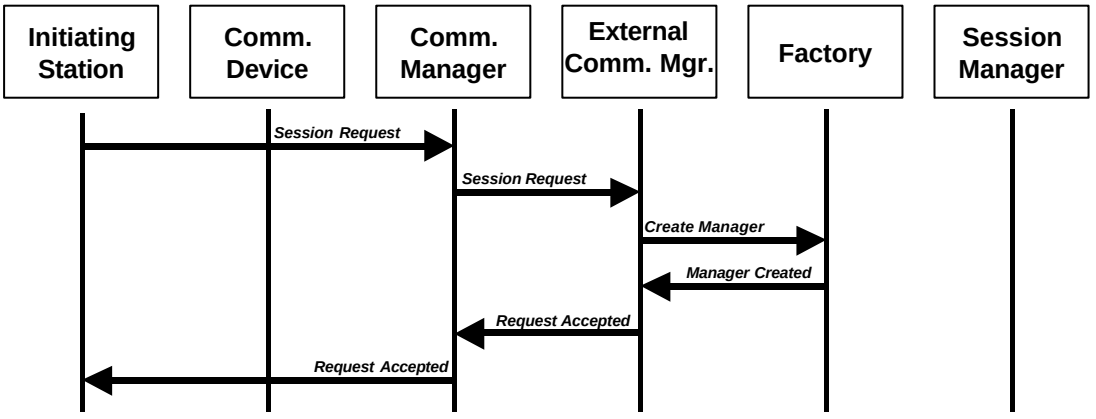


**Figure 68. Station Accepting Request to Establish Session**

#### 3.2.4.2.3 Launching An External Communications Service

At this point, the stations can begin to explore and/or employ each other's services. In the case that the session was established in response to the processing of a caregiver or patient context, the initiating station will attempt to connect to one or more specified resources on the remote station. This is accomplished by first establishing on one or both stations software components needed to proxy interactions between remote components

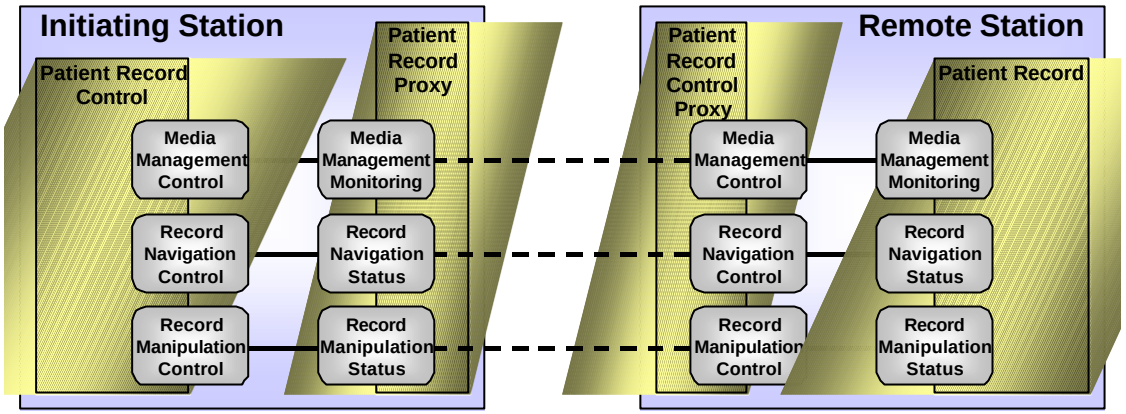


**Figure 69. Use of Proxies in Remote Interactions**

and then establishing the appropriate subscriptions between these proxies and components residing on both stations.

For example, consider the case that the initiating station wishes to interact with a medical record repository on the remote station. The use of a repository implies the use of two complementary sets of controls – the controls embedded in the repository itself and the controls embedded in the components that manipulate the repository. In the figure, proxies of each type appear on each platform as a means of encapsulating communications needed to allow remote interactions. In other words, a proxy exists to allow components on a station to interact with remote components as though they were local.

While this may seem like an overly complex way of dealing with remote interaction, this pattern allows for more complex system interactions than implied by Figure 69. For example, assume that the initiating station has some way of determining that the medical

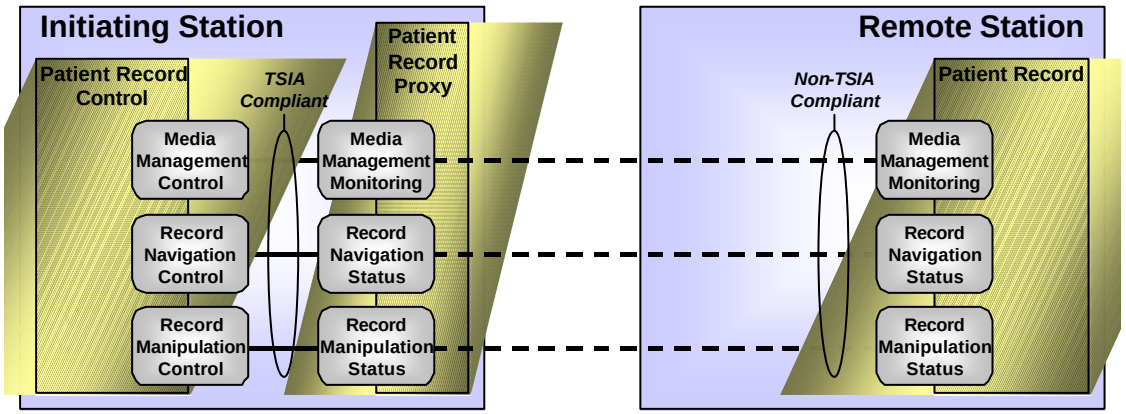


**Figure 70. Use of Proxies in Protocol and Format Translation**

record repository on the remote station operates according to a set of interfaces other than those standardized in this architecture by the telemedicine community. The use of proxies provides a way for converting between this other format and protocol and that used internal to the telemedicine station (Figure 70).

To establish proxies for a specified resource on a remote station, the initiating station pursues the interactions shown in Figure 71 and Figure 72. The protocol controlling this operation first queries the remote station's registry. This is accomplished by using the channel set up through the session managers on each station. When the lease is established, the protocol then directs the local session manager to establish a channel to the leased resource. Working in concert with the session manager on the remote station, proxies are established on each station and interconnected to form the requested channel. In like fashion, the protocol is able to establish communication channels on both the local and remote station's internal communication buses.

Once all of the needed resources have been secured, the protocol knits them together using the standard subscription mechanisms. As with leasing of remote resources, all subscription communications between the local protocol and remote resources are mediated via the session management channel (although this mediation is not shown in the figure below). When this subscription process has been completed, components send and receive messages as normal, irrespective of where they are located relative to one another.

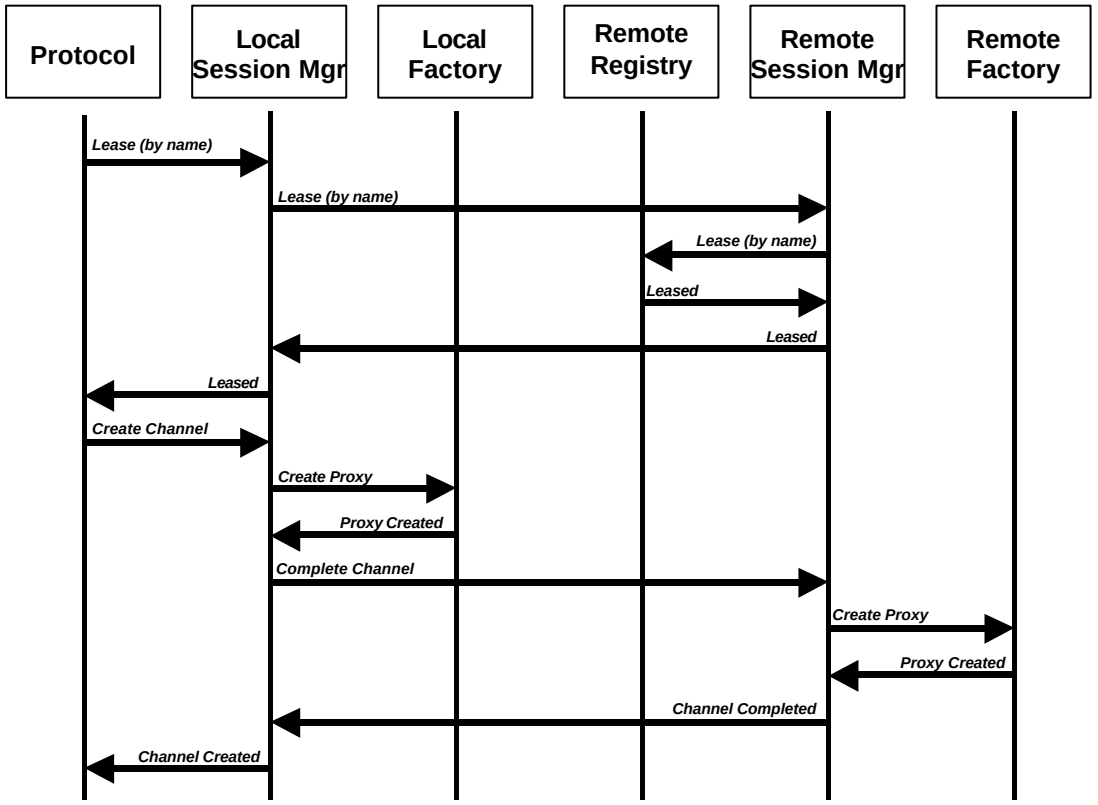


**Figure 71. Protocol Leasing Remote Component and Setting Up Channel**

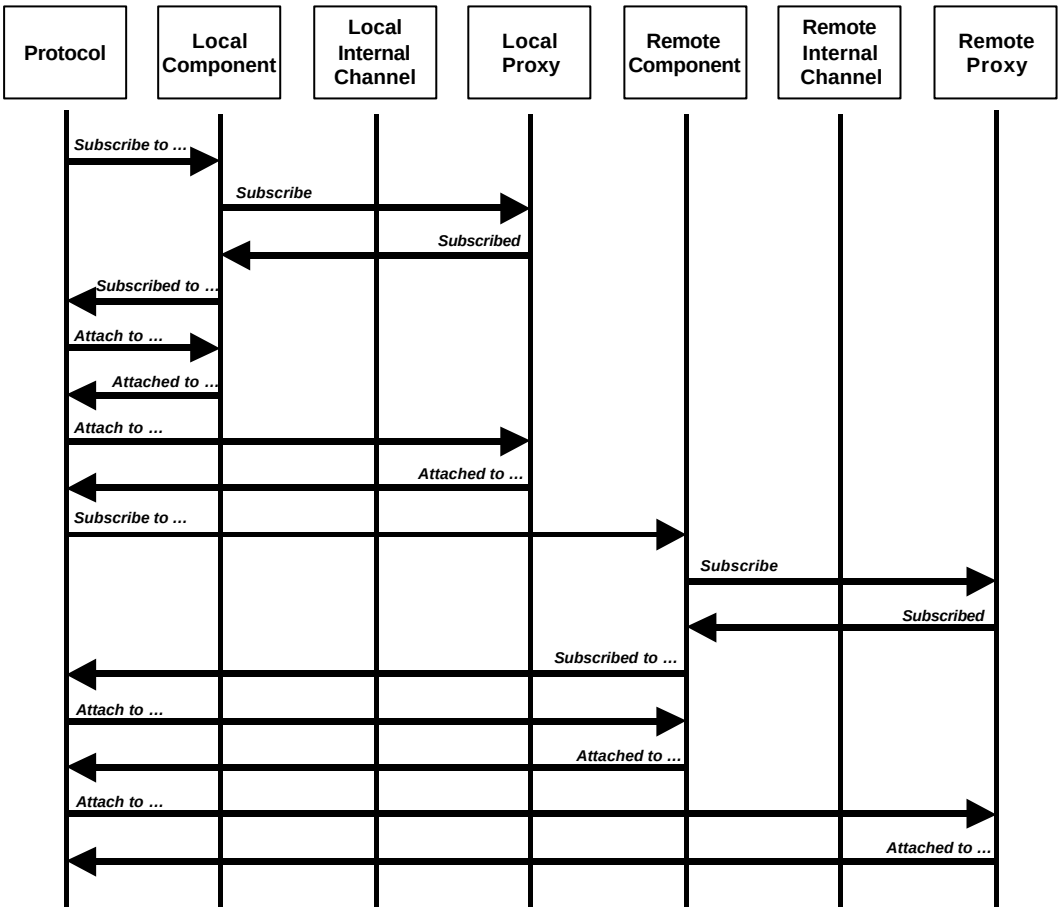


**Figure 72. Protocol Establishing Local and Remote Subscriptions**

#### 3.2.4.2.4 Exploring Services On A Remote Station

As noted earlier in the document, one of the goals of this architecture is to allow stations that have never before interacted to discover each other and to then explore and employ each other's resources. In order to do this, an initiating station must be able to do with the remote station's components many of the same things that it does with its own components. In particular, it must be able to query the remote station's registry to determine what resources are available, to lease components on the remote station, and to cause local and remote leased components to subscribe to one another as needed to support associated protocol operations (Figure 73). As with the remote leasing and subscription interactions described above, all interactions shown in the figure below are mediated by the session management proxies established when the stations first made contact with one another.

Finally, before moving on to a discussions of specific types of external communication services, it should be noted that, while the interactions discussed above were all couched in terms of an initiating and remote stations, there is nothing to say that many of the interactions could not be initiated by the remote station. For example, while a nurse could certainly "call" a patient's station and end up leasing and controlling devices on

this remote station, the patient station could just as easily initiate the call. In this case, it would be the "remote station" that ends up exploring and leasing capabilities on the initiating station. Similarly, it is easy to imagine scenarios in which control of resources local to one station is shared by components both on that local station and on the remote station.

### *3.2.4.3 Notes on Specific Communication-Related Services*

Given these general patterns for managing external communications, there are a number of services that need to be implemented based on these patterns.

#### 3.2.4.3.1 Registry Server Operations

While each station could (and will) maintain its own address book that it would use when needing to contact another station, a more general solution for locating the network addresses of other stations is needed. To this end, the architecture proposes the establishment of a network of registry servers that will serve as a means of finding specific stations or just stations of a given type.

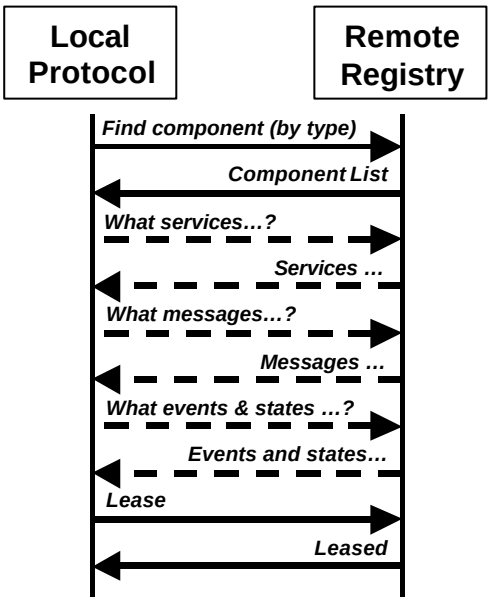


**Figure 73. Station Accepting Request to Establish Session**

When a station first wants to join a telemedicine network, it makes contact with a registry server through the use of a well-known network address. The station is then put in contact with an appropriate registry server in the registry network and a secure communication session is established between the server and the station. The station then passes identifier information (e.g. , station owner, the station's common name, where the station is physically located, etc.) to the registry server, which stores this in the station record that it has created in its database. The server then returns the "station ID", a globally unique identifier that is then stored by the station in its registry. After the

session is completed, the registry server begins propagating the new station data through the registry server network.

If a station that has already been registered moves to another location on the global network (i.e., its network address changes), then the station will notify a registry server of the fact via a secured message. Once accepted, the change propagates through the server network.

When another station wishes to find another station, it passes a query to a registry server and, if the station is found, receives back a network address for the station. This query can be based on the station ID of the target station or on a combination of fields in the server database that uniquely describe the station of interest. Alternatively, if the querying station is interested in finding a given *type* of station (e.g., all home health agencies), then a query containing the search criteria is passed to a registry server and the network address of all stations meeting the qualifications are returned.

A station wishing to withdraw its registration sends a secure message stating this to a registry server. Once authenticated, the server notifies the requesting station and then begins propagating the change through the server network with the end result being that the station's record is expunged from the databases of all registry servers.

#### 3.2.4.3.2 User Authentication and Credentialing

Before remotely located users can access a station's services, they may be required to prove their identity to the station. Likewise, before remotely located users begin to transact business, they want to know the "credentials" of remotely located users. For example, patients may want to know that they are dealing with a given kind of doctor and one who is licensed to provide care. Caregivers may want to know whether the patient with whom they are dealing are insured and by whom and for what. Given these things, the protocols that make up this interface will provide mechanisms for transferring the information needed to support both kinds of operations.

#### 3.2.4.3.3 Session Management Operations

This interface deals with the ability of stations to contact and explore one another and to negotiate how sessions will be conducted. Just as other components in a station must be able to self-describe, session management requires that the station's communication resources be able to describe their capabilities to the communication resources on other stations.

This interface also embodies the mechanisms used by the station to interact with the networks to which it is attached for the purpose of establishing and monitoring quality of service agreements.

#### 3.2.4.3.4 Medical/POCT Device Operations

In support of employing medical devices attached to a remotely located station, the telemedicine system architecture must enable both monitoring of the data being generated by a device and controlling the device's configuration and operation. In the case of monitoring, the key concern (after secure communications) is how to establish the

contextual data related to the readings (e.g., whether the person was standing, sitting, or lying down when the blood pressure measurement was taken).

In the case of remotely controlling, things can become much more complex. In time sensitive control situations, quality of service guarantees are critical. In many devices, the ability of someone to hack into a device and harmfully reconfigure it is a major concern.

#### 3.2.4.3.5 Clinical Imaging Device Communications

Just as the station must provide for the remote monitoring and control of medical and point of care instruments, it must be able to remotely monitor and control the operation of clinical imaging devices.

#### 3.2.4.3.6 Person-to-Person Communications

Protocols that must be implemented in this portion of the station-to-station interface include those that support videoconferencing (to include audio and data sharing services), email, and chat / instant messaging.

#### 3.2.4.3.7 Patient Record Communications

Once a station has discovered that a remote station that it is exploring contains patient records, it has the ability to both read from and write to some or all of these records (depending on the authorizations enforced by the station).

#### 3.2.4.3.8 Dynamic Configuration Communications

If the full potential of this plug and play architecture is to be realized, it must be possible for normal people to assemble telemedicine stations by simply snapping together the set of hardware components that support their particular needs. In assembling such a system, software configuration has the potential of being complex enough to keep us from realizing this goal. Given this, the station itself must be able to determine what software it needs and to automatically find and retrieve this software if it does not exist locally. Controlling this process is the job of the station's installation manager.

When a device is added to a station whose type is unknown to the station, the device bus manager for the bus to which the device was attached notifies the installation manager of this fact. The installation manager then begins the process of finding and retrieving all of the software components needed to support this device by first locating where the supporting software components can be found. This can be done in either of two ways. In the first, the new device stores the network address of the location at which its supporting software can be found. In the second, the ID of the device is passed in a query to a server network (possibly the registry servers) that returns the network address of the location from which the software can be downloaded.

Software components stored at the identified location(s) will be in the form of an archive. To retrieve the archive, the installation manager passes a request containing the device ID to a server, which, in turn, returns the corresponding archive to the station. Once the archive is in the station, the installation manager verifies the security attributes of the file(s) and then begins the process of opening the archive. The components stored in the

archive are extracted, loaded into the station, and then activated such that they announce their presence and begin the registration process.

Based on a standing subscription that the device bus manager has with the registry, the device bus manager is alerted when the device's factory has been loaded onto the station and, in response, it triggers the factory to generate a proxy for the device. As with all such proxies, once it is created, it harvests the needed information from the device that it represents and then registers itself with the station.

## 3.2.5 Terminologies and Concept Representation

The exact details of which vocabularies to use and which representations will necessarily be delayed until other aspects of the architecture are settled out. At the same, it is worth noting that two different strategies might be employed here. The first is to standardize on a telemedicine-specific "native" language and encoding for station-internal operations and to translate between this native format and other standards when needed. The second is to try to come up with a spanning terminology set that is drawn from the requirements imposed by the different component standards embraced in the ultimate version of the specification.

## 3.2.6 Assuring Station and System Operations

In the world assumed by this architecture, a number of key security and safety issues will need to be addressed. While the specific details of how to secure an individual station and a system of stations will be dependent on the specific technologies finally chosen for the implementation of the architecture, an number of things can be said at this point about the kinds of concerns that must be addressed and about potential approaches to addressing these needs.

### *3.2.6.1 Assurance Concerns*

As distributed computing systems, telemedicine systems and the stations of which they are composed face many of the same problems as distributed systems used in other applications. As medical systems, they must address the same kinds of issues faced by other kinds of clinical devices. Among these are:

- control of patient data,
- control of access to system services,
- dependability of services,
- auditability of services, and
- issues related to operational context.

#### 3.2.6.1.1 Control of Patient Data

Control of patient data falls into three broad categories: enforcing privacy requirements, guaranteeing its accuracy, and ensuring its availability when needed. With respect to privacy, the "bare" architecture discussed so far in this document presents a number of concerns. First, data stored in repositories on a station might be read directly from the records if no access controls are in place. Likewise, the files in which the repository data is stored might be copied. The media (e.g., patient cards) on which these files reside might be taken and read directly if not somehow protected.

Data moving along the various communication links within a system are also open to unintended disclosure. Eavesdroppers can intercept communications between stations. Traffic flowing between components within a station can be picked up by unauthorized users, especially in environments in which multiple stations exist and share resources from a common pool of components. Even communication between stations and "peripherals" (e.g., a medical device or a user interface on a PDA) are vulnerable to interception, such as when RF links are used to connect the communicating devices.

A more subtle concern is persistent storage of information in components used by a community. If, for example, a user interface component caches data sent to it by the rest of the station, it might be possible for data generated in one user's "session" to be available to the person using the component in the subsequent session.

An even more subtle issue is the use of inferencing to learn things about a person that are otherwise safeguarded against disclosure. One example of this would be tracking who is talking to whom. While the observer may not be able to directly read any of the traffic between a patient and his caregiver, mapping calls or network connections might be enough to infer what kinds of health issues the patient is facing. In the discussion that follows below regarding a security architecture, this issue will not be addressed. One way of approaching this in the future is through the use of trusted intermediaries that proxy users to outside entities.

Even in cases where a given individual or group is granted access to data handled within a system, there will be times when this "authorization" may need to be limited to only a subset of the data within the system. For example, a patient might be authorized to read his "official" encounter records contained in his caregivers' records but not the "personal" notes maintained by the caregivers. In addition, access to a record given to an individual in some circumstances may need to be restricted in others. A nurse may have access to a patient's chart during work hours and while at the hospital but not be permitted access outside of hours or from off-site.

As with privacy, the integrity of patient data must be addressed on several fronts. In communication in its various forms within a station or system, data elements can be corrupted. If the threat is passive (e.g., noisy or unreliable communications), then simple mechanisms for detection and correction are appropriate. If the threat is active, then more advanced (e.g., cryptographic) techniques are needed. In storage, data can be corrupted at each of the levels described earlier. Individual records can be corrupted. Files can be altered independent of the mechanisms used to deal with records internal to these files. The media on which the files reside can fail. In the case of the first, in as much as medical documents are signed for the purposes of attribution and attestation of veracity, steps must be taken to ensure that "signed" records in a repository can only be amended according to rules of accountability.

Finally, control of patient data implies that those users with a legitimate need to access patient records must have assured and timely access to this data. In addition to

redundancy mechanisms (such as backup and restoration of records or replication across multiple repositories), this means that it must be easy to locate and rapidly retrieve relevant information. In some cases, this may mean providing mechanisms for allowing certain users of the data to override default access control mechanisms when the timeliness of access is exceptionally critical.

#### 3.2.6.1.2 Control of Access to System Services

The ability to control how and when system and station services are accessed and by whom has implications for both security and safety. As a starting point, it is necessary to control how "authorized" users employ a system's resources. Not all functions are appropriate for all users. Certain clinical functions (especially those that are safety critical) may need to be off-limits to patients being treated with the devices that deliver these functions. Likewise, certain maintenance functions, such as device calibration, may need to be restricted to personnel trained in these functions.

In telemedicine systems, it is entirely possible that a device's functions may be accessible both remotely (i.e., controllable by some remote device) and locally (e.g., through a front panel on the device). In these cases, it may be desirable –even necessary – to allow the device to be configured to preclude the use of one method of control or the other. In these sorts of cases, care needs to be taken to ensure that access control cannot be subverted simply through replacing one component with another (e.g., by replacing one interface device that has been configured to enforce interface function access controls on a station with another like device that has not been configured in this way).

In some cases, it will be important that a station's configuration remain inviolate. In these cases, all actions having to do with station configuration, such as addition of new devices, must be rejected by appropriate station elements, such as the registry.

Even when remotely generated commands are encrypted or simply cryptographically signed, steps must be taken to guard against "replay attacks". In these attacks, valid commands are recorded by a "man in the middle" and then replayed to and acted upon by the target device that sees the commands as valid.

#### 3.2.6.1.3 Dependable Services

If clinicians and patients are to use telemedicine systems to carry out healthcare transactions, they must be able to trust that these systems will act faithfully and predictably on their behalf. Of primary concern will be the correct operation of remote devices and their correct employment. It is easy to imagine devices from a range of manufacturers delivering a given agreed upon body of functionality at different levels of fidelity. In addition, certain kinds of devices require periodic calibration and maintenance. In both cases, caregivers employing these remote devices will want to know the degree to which the data from these devices can be trusted for the clinical task at hand. Related to this issue is the question of whether or not the person manipulating the device at the patient's location is operating it correctly. For instance, correct interpretation of EKG traces depends on the leads having been correctly emplaced. In cases of remote operation, a caregiver may be hard pressed to know with certainty that this has been done. While certain technical approaches might be used to address this last

issue, the central issue is operator expertise and reliability and no attempt will be made in this document to suggest how this problem might be solved.

Much of telemedicine is about reengineering of care delivery structures – changing where things are done and by whom. In order for this sort of restructuring to be effective, devices made for remote operation must be designed for use by lesser-trained caregivers or by non-clinicians. This may necessitate incorporating safeguards into such devices that all but eliminate the possibility of the devices being used incorrectly or that limit the damage that can occur from such misuse. Once again, how this is to be accomplished will be addressed in this paper.

Because telemedicine systems are often employed in less than ideal settings, communications reliability can be a significant issue. Consequently, systems intended for use in these environments must ensure fail-safe operation in the face of communication system interruptions. In addition, in performance critical operations, it must be possible for the stations involved to gain a sense of whether or not the associated infrastructure can guarantee certain responses before the stations commit to a given course of action.

Another key issue in the area of dependable services is the question of system composition. One of the goals of the architecture is permit ad hoc, plug-and-play composition of stations from components produced by a range of developers and to allow systems to be composed of stations that have never before interoperated. Whereas current systems are certified by organizations like the FDA as turnkey capabilities, mandating this approach to certification of future telemedicine systems would gut the benefits to be realized by a component-based station architecture. Some other means of ensuring that arbitrarily composed stations and systems are safe must be found. Related to this is the question of how a station knows that it can trust the software that is downloaded into its space when it is automatically configured to support a new device.

#### 3.2.6.1.4 Auditable Operations

Much of what has been discussed so far in this section is predicated on the ability to strongly authenticate all parties and system elements to one another. In remote operations, caregivers must know who is on the other end of a link. They must have a sense that any data generated by a remote station during a session or actions taken by that station are associated with the person that they believe to be on the other end of the link. Likewise, when patients "visit" a remote caregiver, they want to be able to identify the individuals with whom they are dealing.

In many cases there will be a need to log what operations were done using a given station configuration, when these operations occurred, who performed the operations, on whom they were performed and with what resources. In addition to the obvious connection to patient records, this audit log feature may be required for, among other things, tracking down erroneous data generated by a faulty component or for identifying operator problems that need to be remediated in some way.

#### 3.2.6.1.5 Issues Related to Operational Context

A variety of assurance measures are available to designers of sure components and systems but not all measures are appropriate to all uses. Likewise, different countries have different laws regarding what kinds of mechanisms can be used for a given need. As a result, it may be difficult for the telemedicine community to dictate a specific suite of mechanisms for use in safeguarding stations and systems. For this reason, it will very likely be necessary for stations to explore each other's assurance capabilities and to negotiate an assurance "policy" that dictates what needs to be secured and how during the course of a session conducted by these stations.

Finally, even though much of what is discussed in this document focuses on interactions between two stations, nothing precludes multiple stations interacting with one another at the same time. One example of this would be a three-way conversation between a patient, his caregiver, and a consulting specialist. Given this, solutions chosen for securing station-to-station interactions should be capable of networked operation and not just point-to-point operation.

### *3.2.6.2 The Trust Model*

In securing a system, the chief question is always "In what are you placing your trust?" Note that in security, as in most things in life, there is no silver bullet that can solve all problems; therefore, when we say that we are trusting some portion of the system, the salient question is "For what are you trusting it?" Figure 74 highlights those parts of the system architecture in which it is proposed that trust be placed for some aspect of security.

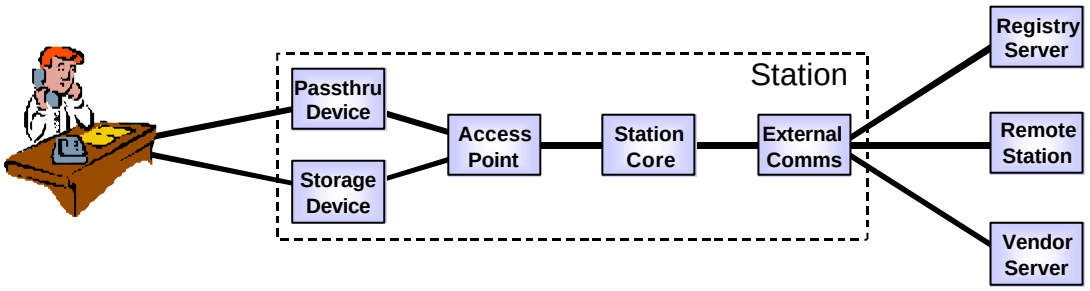


**Figure 74. Points of Trust in Architecture**

In brief, here are the roles that each portion of the station and system play in maintaining ensuring secure system operation.

#### 3.2.6.2.1 Humans

Human operators can play any or all of four roles: station user, administrator, maintainer, or auditor. ***Station users*** perform all of the operations described earlier in this document (adding, removing, and operating devices and employing other station functions). ***Administrators*** establish authorizations for all other users, specifying what they can and cannot do on a given station under specific conditions. Under certain circumstances, they manage the configuration of stations and devices, controlling what kinds of devices or

which specific devices can be added to a given station and establishing specific settings for both. ***Maintainers*** have responsibility for ensuring that equipment is operating correctly. In some cases, this will entail employing device functions unique to this role and configuring certain aspects of device operation. Finally, ***auditors*** manage audit logs maintained by stations. This management includes setting authorization for reviewing, editing, archiving, and deleting audit logs. The goal in separating auditing from administration and maintenance is to allow for independent review in environments where this level of oversight is required. Note that, in this discussion, roles and individuals are not the same thing and that any one operator in a telemedicine system may be empowered to play more than one role, up to and including all roles. How the assignments are made will depend on the security concerns associated with each station's and system's operating environment.

From a trust perspective, the system security architecture relies on all four user types to protect their user authentication mechanisms (e.g., passwords or security tokens). Except where independent auditing is employed, it also relies on each authorized user to safely and securely use the capabilities to which they have been given access (e.g., to not employ devices in ways that would harm a patient or to access records only on a need-to-know basis). Administrators are trusted to correctly implement authorizations for all system users. Auditors are trusted to regularly review and analyze audit logs to assure that authorized personnel are behaving as they should and that unauthorized users are not accessing a system's services.

#### 3.2.6.2.2 Pass-through Devices

These are those devices in the system that exist to move data between the cyber and physical worlds (e.g., medical devices, user interfaces devices). These devices will come in two flavors: untrusted and trusted. Untrusted are those suitable for use in environments in which access to stations is controlled and in which the devices may be added to or removed from a station only by authorized personnel. In these environments, it is also assumed that all device bus communications are protected by the environment (i.e., no one can eavesdrop on private data streams or spoof either the devices with bogus commands or those elements that would use the devices with bogus data that appears to come from the device in question). Note that these assumptions are being made today in prominent medical device standards, such as IEEE 1073 and the POCCIC lab device interoperability specifications.

Trusted pass-through devices are those meant for use in inherently untrustworthy environments in which access to stations is not tightly controlled and in which the device buses may be accessible to unauthorized parties (e.g., when RF communications are used). At a minimum, trusted devices are assumed to maintain a universally unique ID and to provide a cryptographically secure means of ensuring private and authentic communications. At a minimum, these secure communications protect the data flowing across the associated device bus. Ideally, they would support negotiation and implementation of both encrypted and cryptographically authenticated data communications between the device and other components within a station, such as the registry, authorized protocols, and end components that subscribe to the device and to which the device subscribes. These devices are also assumed to not rely on a proxy for

implementation of their security capabilities and, instead, implement these functions internal to themselves. Ideally, these devices also carry some sort of tamper indicating mechanisms that make undetected alteration of their internals difficult to achieve.

#### 3.2.6.2.3 Storage Devices

This category includes both those devices used to store medical records and those used to store patient context information. If the medium on which files and other data items are stored can be physically protected, then only access controls on the underlying file services and on the overriding record services are required to ensure the privacy of these records. If the medium can be removed, then additional requirement of data encryption may be levied on the device to ensure that data is protected when the medium has been removed from a station. Also, in both cases, some form of data authentication may be required to ensure that data, once stored, cannot be corrupted without detection. Lastly, storage devices and, therefore, the data that they support are always vulnerable to single point failures. For this reason, some sort of redundancy in storage is always recommended, whether this be through periodic backup of files, shadowing of files, error correcting mechanisms that allow partially corrupted files to be reconstituted, or some combination of the above.

#### 3.2.6.2.4 Access Points

As drawn in the figure, "access points" are communication platforms that might be used in distributed station designs and that serve as bridges between device buses and a station's internal communications bus. A component of this type would typically contain one or more device bus managers (depending on the type and number of bus transceivers that the component incorporates), proxy factories for each kind of device that the device buses might host in a station, device proxies, and whatever "channels" are established to allow other communication devices to interact with proxies that have been launched on the component. From a communication security perspective, these components can operate in either of two ways: unsecured and secured.

In the unsecured mode, these components pass unaltered all traffic flowing on both the component's device buses that they host and on their connections to the internal communications bus. This mode may be used in environments in which physical security measures are assumed to provide adequate protection to a station's internal communications or in environments in which trust relationships are established between the devices hosted by the component and devices located elsewhere in the station. The first case, the component is routing unencrypted traffic; in the second, the traffic that it carries is encrypted.

In the secured mode, the component manages the encryption and decryption of traffic moving onto and off of the internal communications bus to which it is attached. This mode allows for creation of stations in which station resources, such as a medical record repository or an external communications gateway, may be situated at a different location than the medical and user interface devices attached to the component collocated with the station user. While the device bus communications may be considered to be secure based on constraints placed on its physical environment, the larger internal communications bus is not.

Beyond encryption, an access point may be trusted to support certain aspects of access control. In particular, the process of registration and subscription described earlier in this chapter did not address the question of who was adding or removing components from a station. In a station that implements access control for each of its services, it must possible for the station to be configured to only accept devices being installed by a specific station user, to only accept specific devices (i.e., devices having specific unique IDs), to only accept specific kinds of devices, or some combination of the three. In the case of an access point to which the device is being attached, this means that the interactions shown in Figure 54 and Figure 38 now function together as shown in Figure 75. Once the registry has refused to add a device to a station, the proxy for that device is terminated.

#### 3.2.6.2.5 Station Core

The station core shown in Figure 74, represents the station's central building blocks – in particular, the registry, the default context, the internal communications bus' bus manager, the external communications manager, and the installation manager. Taken together, these components will have central responsibility for governing access to station resources

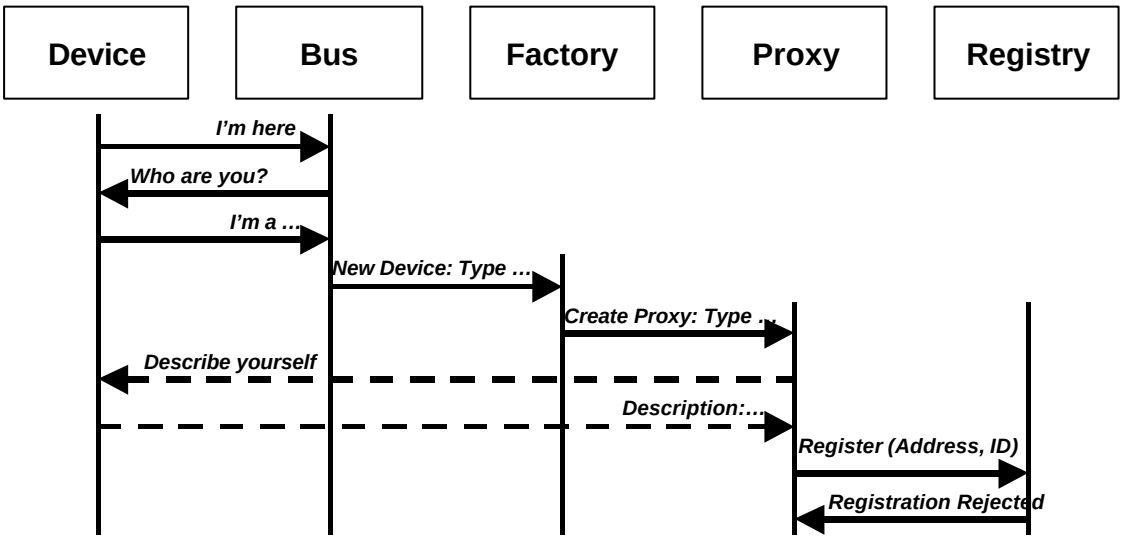


**Figure 75. Refusing the Addtion of A Device to a Station**

The station core is also trusted to support secure station-internal communications (encrypted and/or authenticated) where needed. As with the other components given this responsibility, implementation is achieved via mechanisms built into the channels that the station core employs.

Next, the station core has primary responsibility for logging safety- and security-critical events along with other events of interest identified by the station's auditor(s) and administrator(s). The station core is trusted to ensure that these records are kept private

(i.e., divulged only to those users with proper authorization) and available (i.e., like other storage requirements, protected against single point failure of the underlying storage medium). Note that this auditing requirement implies the presence of audit-enabled services in the station's other components.

#### 3.2.6.2.6 External Communications

The station is involved in three sets of external communications: communications with other stations, communications with registry servers, and communications with software servers. In each of these, the external communications channels are trusted to ensure, as required, privacy and/or authenticity of data crossing these links. As with station-internal communications, this trust is vested in the mechanisms that implement the "channels" set up in support of external communications.

Both to support auditing and to ensure the integrity of medical records, stations may require the use of a trusted external time source. Depending on the approach used to implement this requirement (i.e., either synchronizing internal clocks with the external or retrieving time stamps from the external source) the external communications will be trusted to ensure accurate synchronization with the external source or to support generation of cryptographically-provable binding of station-internal records to externally-generated time stamps.

#### 3.2.6.2.7 Remote Station

As the access point for remotely located users wishing to employ the local station's resources, remote stations are trusted to enforce authentication of all users of those stations.

#### 3.2.6.2.8 Registry Server

In station-to-registry server communications, both the station core and the registry sever must provide mechanisms that ensure that only the station can ever present itself to the registry server as that station. This implies that once a server assigns a unique ID to a station in that station's initial registration process, the registry server (and all of the other servers in the network) are trusted to never again issue that same ID and to protect the integrity of registration data.

If secure time-stamping is used, the registry servers are logical candidates for providing the external time source data. This would mean that the server network would trusted to ensure the integrity of time-stamp data that it provides.

#### 3.2.6.2.9 Vendor Server

In station-to-vendor server interactions, the device vendor is trusted to have secured the downloadable software in a way that allows the station's installation manager to verify the authenticity of the software package's source and the integrity of its contents. The vendor is also trusted to have ensured the integrity of the code prior to its packaging for distribution.

### *3.2.6.3 A Security Architecture*

Given the system assurance needs and the trust model elements described above, the following security elements are proposed for the telemedicine system architecture.

#### 3.2.6.3.1 Trusted Platform

The foundation of the station's security is the ability of the platforms that host the various station processes to ensure that these processes cannot be corrupted, either when running in program memory or when in longer term storage (e.g., disk or ROM). In as much as one of the architecture's goals is to allow for code mobility, so that processes can be run from the "right" place in a given system, this foundational element requires that the code modules carry signatures that allow their trustworthiness to be examined before being loaded in to program memory.

#### 3.2.6.3.2 Secure Station-Internal Communications

Assuming that the processes running on a given station platform are secure, the next element required in the security architecture is a means of knitting together processes scattered across independent platforms. For example, a user of a home system may have a set of external communication processes that run on a home gateway, the registry and other core station processes as well as data storage processes that run on a network-attached general purpose computing platform, device-specific processes that run on a network access point, and user interface processes that run on a net-ready palmtop PC. In instantiating a station, each of these components may need to be securely connected to one or more of the other components at some point in a session.

Establishing secure, station-internal communications is a two-phase process. In the first phase, preliminary information needed for eventually establishing a secure link is created and positioned where required. This can occur in either of two ways. In the event that a system's composition is essentially static (e.g., as in the components that might be found in a home care setting), then the initial "installation" process that is followed when a device is added to the station for the very first time will include a step in which all devices already attached to the system and the new device are all told that it is okay to trust each other. Depending on the particular approach ultimately selected by the telemedicine community, this may result in the sharing of some common secret (e.g., like the "base" number used in Diffie-Hellman exchanges). In the event that a station's composition is quite dynamic (as in the case of a doctor's PDA moving from room to room during rounds or equipment on a ward that is taken from the supply closet and placed in rooms as conditions dictate), then an alternative mechanism (such as the creation and registration of certificates) is used.

Either way, once the initialization has been completed, devices are ready to establish secure communications whenever required. Whereas, in other applications, this might entail the entities that want to interact first exploring each others capabilities and then jointly agreeing on a policy to be pursued for the session, in this architecture, each entity's capabilities will have been part of the information harvested from the entities when they first registered with the station and the problem of deciding what capabilities to employ will fall to the protocols that lease these entities. Given this, all that remains

for the entities is to establish the secured session in whatever fashion has been dictated to them using the information established during initialization as the starting point for this task.

#### 3.2.6.3.3 Secure External Communications

Because a key goal of the architecture is to allow stations that have never before interacted to discover each other and to then begin transacting business, independent stations will employ certification (e.g. public key infrastructure) approaches as the basis for bootstrapping trusted communications. Because a given station can host multiple simultaneous communication sessions, with each possibly having its own approach to assurance, implementation of each security mechanisms will be embedded in the external communication channels set up to support these sessions. While more complex than some other approaches that might be taken, this allows the station the greatest flexibility in supporting existing standards that may have already selected their own (potentially disparate) approaches to securing their protocols. As with station internal communications, the security policy used for a given session (e.g., "We will use encryption algorithm X with key of length Y established using key exchange algorithm Z and we will use counter-based anti-replay starting with counter value A.") may be established dynamically based on negotiation following a period of mutual exploration of each other's capabilities.

To address quality of service issues, the telemedicine community will need to select a suite of protocols that allow individual stations to interact with the communications infrastructure to which they are attached with a view to negotiating quality of service guarantees from this infrastructure. Similarly, protocols will need to be in place that allow the stations to monitor network status and to be alerted when the network believes that it will not be able to support the existing QOS agreement. While much work is going on in this area in the networking community, much of this work is still nascent and the telemedicine community may need to wait a while before committing specific protocols that address these needs.

#### 3.2.6.3.4 User Authentication

Authentication of users will be handled in two ways, depending on where a user is relative to the station of interest. For a user logging into a local station, the station will provide a "trusted path" between the interface mechanism used to present user identification and authentication (I&A) data (e.g., user name and passwords or token-based data) and the processes in the station core that handle validation of this data. In situations in which the user interface devices are integral to the platform on which the core elements are running, this trusted path may be provided by the core services and/or the foundation (e.g., operating system) on which they rely. In the case of peripherals (e.g., a PDA or security token attached to a network access point, this path may consist both of secured processes running on the trusted station platforms and secure communications established from the interface point back to the station core. In either case, it is the core that accepts or rejects the data presented and that notifies the access control processes in the core when a user has been logged onto the station.

When the user accessing the station is remote, several different approaches are possible depending on how the telemedicine community wants to handle this issue. In the first, each station handles remote users in much the same way as local users (i.e., with authorization tables and I&A data being stored by the station). In this case, the I&A data might be passed from the remote station over a secure link established by the remote and local stations. Alternatively, a one-time password mechanism might be used in lieu of a secure channel as a means of protecting against passwords being captured.

In the second approach, a global certification system that allows any user to be identified and authenticated to any other user is established and users are able to make authorization decisions regarding what services (e.g., access to patient records or the ability to monitor and/or control devices on the station) if any to expose to given remote users either on a session-by-session basis or using standing authorizations as is suggested in the first approach. .

The third approach uses either the global certification scheme of the second or the station-centric approach of the first but uses role-based controls rather specific identity-based controls to mediate access. In this approach, a station grants privileges to classes of users who can prove that they belong to a given authorized class.

Of these approaches, station-centric methods are the least favored. in that they only work well for small, closed communities whose user populations do not change much or often. Of the remaining two, role-based allows some degree of simplification in administration and identity-based provides the potential for finer-grained controls.

#### 3.2.6.3.5 User Credentialing

Related to the question of user authentication is that of credentialing. In an environment where interactions may be conducted only over the wire, some means of establishing what a person is capable of doing and authorized to do. In the case of caregivers, this can include, among other things, the ability to pass to remote users information about licensing and board certification. In the case of patients, this might include information regarding insurance.

Addressing this need involves the use of certification of participants in a session and external certification servers. In particular, when user stations establish a session, they can pass requests to each other for credentials. In response, certificate containing the credentials are exchanged. Embedded in the certificates will be an indication of where the receiving station can go on the Internet to ascertain the authenticity of the certificate. In typical PKI fashion, this chain can be followed as far as required to reach a certifying authority that the interacting stations share in common.

#### 3.2.6.3.6 Access Control

As discussed above, access control expresses itself in at least three ways in the proposed telemedicine station architecture:

- control over how much a station's capabilities a given user is made aware of,
- control over which information and services a given user can access, and

- control of a station's configuration.

In the first case, the principle pursued is one of "least privilege" – a user accessing a station either locally or remotely will only the minimum essential set of services and information implied by the authorizations established for that user. For instance, a normal station user who has no administrative privileges will not only not be given access to administrative functions, such as establishment of user authorizations, but he will also not even be told by the station that these services exist. The user interface mechanisms that enable control of these functions will be completely hidden from view, as will all of the infrastructure (e.g., files, active processes, etc.) that support these functions. If means exist that would allow the user to attempt to activate these services (e.g., a command line interpreter facility), the station will respond to such attempts in exactly the same manner as it would respond to employ completely non-existent functions.

In the second case, each service (including those related to read and writing patient record data) mediates all user transactions through access control mechanisms associated with the service. For example, if a station user has the right to read a given set of records, the station will allow the user to navigate through the associated record index to specific records and to then open these records for review. At each step along the way, the service will examine any new entities being engaged and, based on the user's authorization settings, decide which actions related to those entities can be offered to the user.

In the final case, any given station can establish constraints on how the station can be configured in the presence of a given user. The effect is to set bounds on what degree of customization can be done either through the use of a user's context or through manual configuration of the station (e.g., addition of new devices) by that user. An example of the first would be a station rejecting a context-driven directive to stitch a given external processing capability into the station or to use a given set of externally hosted protocols to drive station operation. An example of the second would be the station refusing to accept the attachment of a lab instrument that the user is not certified by the station's owning organization to operate.

As noted earlier, in all cases, decisions regarding what to allow and what to reject are centralized in the station's core but enforcement may be carried out in other places. One example of this would be an access point refusing to establish a proxy for a device that has been attached to its device bus.

#### 3.2.6.3.7 Authorization

Processes in the station's core own authorization information established by a system administrator. Specifically how this information is safeguarded is dependent on the nature of the platform itself (e.g., an embedded implementation might store the data as encrypted records written to non-volatile memory whereas a PC-based implementation might use the default mechanisms resident in the computer's operating system).

Given the wind range of settings and ways in which users may want to use telemedicine systems, it is suggested that the telemedicine community not constrain the kinds of criteria that can be used in determining whether a given user is authorized to access a given resource. Indeed, some in the general healthcare informatics community have advanced authorization schemes that are quite sophisticated (e.g., "User X can access these services during this time of day during these days of the week but only if these attempts to access are being made from these particular locations or this particular station."). At the same time, since one of the goals is to enable wholly uninitiated users to assemble and operate their own stations, it is suggested that mechanisms be put in place to ensure that certain minimum "default" authorization requirements are established by the user (e.g., no a priori access is granted to remote user and access is granted in real-time on a case-by-case basis by the station's owner) or an agent of that user.

#### 3.2.6.3.8 Device Certification

One of the questions that caregivers employing remote devices must ask is, "Why do I believe that I can trust the data being generated by a given remote device?" In "normal" clinical settings, much of the trust that a clinician places in a device is based on the ability to physically inspect the device ("It looks to be in good shape – not banged up at all"), on an on-going history with the device ("we used it with every patient that has walked in here today"), and on established quality assurance processes ("The sticker shows that the techs calibrated it just last week"). In current approaches to telemedicine, these same mechanisms are generally still in effect (i.e., the "local" clinician is able to attest to the remote caregiver that these things are so); however, in situations in which these approaches to developing trust cannot be employed, something else is required.

To address this need, the following either of two approaches is recommended. In the first approach, devices are de signed from the start with non-traditional settings in mind. Jostling of the device, splashing with liquids, use in dust-prone areas, and storage in hot or cold places are all accounted for in the design processes such that the devices continue to operate as required even in the face of these adverse conditions. This approach is then coupled with periodic mandatory maintenance that includes testing and recalibration, if required. As part of this activity, the maintainer interacts with the device so as to install in the device a certificate that, at a minimum, indicates when the device was serviced and by whom, and that, ideally, characterizes the device's performance in a way that allows processing routines on a remote station to inspect the performance data and use it, if required, to properly interpret readings generated by the device. In the alternative approach, devices are capable of thorough self-test. In this case, devices report on their own operating characteristics and able to determine when they are no longer able to support their intended functions.

#### 3.2.6.3.9 Secure Time Stamp

To support the need for accurate time stamping of patient records data and audit log information, a telemedicine station will rely on a trusted external time source and a clock function that is part of the station core. Through the use of the appropriate protocols, the station synchronizes its internal clock with the trusted external reference. In the process, the station tracks its own accuracy (i.e., by how much the station's clock had to be

adjusted) and uses this information to develop a schedule for updating its clock such that the deviation from actual time is never outside of some specified tolerance.

#### 3.2.6.3.10 Secure Configuration

In downloading to a station new software components that support a given device, two key concerns must be addressed. First, its supplier must have produced the software in a trustworthy fashion. Second, it must not have been corrupted since its creation. While the telemedicine community certainly has an interest in the first, the question of software trustworthiness is an exceedingly difficult problem that has yet to be adequately addressed by anyone. All acceptable approaches used today are manpower-intensive and, therefore, expensive; and even the best of these approaches cannot assure a flawless product. For this reason, it is strongly recommended that the telemedicine community not "punch this tar baby".

By contrast, the second concern can be addressed by available technologies. In particular, a body of software to be downloaded to a station should contain a manifest that lists the contents of whatever package is being downloaded and that cryptographic authentication mechanisms be used to bind the contents and the manifest to each other and to a certificate identifying the vendor. Packages of this sort received by a station as the result of a request from the station can be authenticated using the vendor's public key drawn from the public key infrastructure used by the telemedicine system. Once authenticated, individual software elements can be extracted and installed by the station's installation manager and then activated such that the components that they support are registered with the station.

In serving up packages to requesting stations, a vendor server may operate in either of two ways. In the first, it simply returns packages that have been packaged well in advance of a request for the package. In the second, the server creates a package once a request has been received. The advantage of this latter approach is that it allows the telemedicine community to choose a protocol in which the station describes its operating environment and the vendor server is able to create packages that meet the unique needs of the station. From a security perspective, this latter approach is more challenging inasmuch as it requires that the vendor server be free from invulnerable to compromise. The concern here is that an adversary who is able to subvert the server will be able to install malicious code into a package and that the station, trusting the digital signatures used to seal the package will install the corrupted code without questioning its trustworthiness. By contrast, the first approach allows for off-line creation of trusted packages, leaving the server to do nothing more than select and return the right package for a given request.

#### 3.2.6.3.11 Secure Registration

As described earlier, when a station first wants to join a telemedicine network, it makes contact with a registry server through the use of a well-known network address and is then put in contact with an appropriate registry server in the registry network. To enable the establishment of secure communications, public key-related information is stored in the station's core by its developer at the time of manufacture. Once the station and server have established a secure communication session. Identifier information (e.g., station

owner, the station's common name, where the station is physically located, etc.) passed from the station to the registry server is digitally signed using the station-specific private key embedded in the core by the manufacturer. In like fashion the "station ID" (a globally unique identifier) returned to the station by the server, is signed using the server's private key and, once authenticated, is stored securely in the station's registry.

In the same way, all other communications within the registry server network will rely on authenticated communications in transferring information between nodes in the network.

#### 3.2.6.3.12 System Security Administration

Implied in the discussions of a number of the above topics is the idea that a station maintains a set configuration data along with a list of individuals or types of individuals authorized to use the station. Managing these things falls to the station's administrators.

With respect to station configuration, the centerpiece is the default context discussed earlier in this document. While this may simply be a single description of what components are needed and how they are to be assembled to create the root station capabilities, station administrators are also free to establish user-specific contexts that describe what is to be done as a matter of course when a given individual or particular type of user logs into the station. In addition to stating what is allowed, this configuration information can also include statements of what is to be prohibited (e.g., a given user is never to be allowed to configure certain aspects of a station in a given way).

That this can be done implies that the administrator is able to enter into the station a list of known users or user types. This required facility will be embedded in the station's core.

In addition to describing what each such user can do regarding station configuration, the system is administrator is also responsible for establishing the authorizations described above.

Finally, there will be times that human operators will need to interact directly with a station's registry (e.g., to reset a field or to change a device's installation status from "permanent" to "temporary"). These tasks also fall to the administrator.

System administration may be done locally or remotely. In the latter case, the establishment of secure communications is presumed.

System administrator functions are embedded in the station's core. As with user authentication, interactions with the core are conducted through a "trusted path" to the user interface devices on the station from which the administrator is operating.

#### 3.2.6.3.13 Auditing

Implementation of auditing is established through the use of subscriptions. As a station component, the audit log (part of the station's core) is able to subscribe to messages associated with other components' internal state. Certain messages that a device can generate will be marked as auditable. Whenever a device is added to a station, the audit

log automatically subscribes to some or all auditable messages assuming that no other configuration changes are made to the station. Through the use of administration and auditing protocols, these subscriptions can be altered such that whenever a specific kind of device is added to a station, the audit log will subscribe to a particular set of messages for that device (i.e., the defaults are overridden and some defaults may be ignored and some message not normally audited will be recorded). The implication in all of this, of course, is that the telemedicine community, in establishing its standard device interface, also establishes which device communications and which internal state changes constitute the default auditable events.

Users assigned the auditor role for a station are able to review and delete audit logs entries as desired. As with other station functions, auditor-related functions can be accessed by authorized users operating locally or located on a remote station. In the case of remote operations, mechanisms for encrypting and authenticating communications in both ways are required. While there does not seem a compelling reason for the telemedicine community to specify them, it is expected that developers will provide tools that aid auditors in filtering audit records according to certain criteria (e.g., events occurring when a given operator was using the station or records related to a particular station component).

Like the system administrator functions, auditor functions are embedded in the station's core and interactions with the core are conducted through a "trusted path" to the user interface devices on the station from which the auditor is operating

## *3.3 Implementation Options*

Having addressed the logical organization and operation of the proposed architecture, the next question to consider is exactly how to render this architecture. To this end, this section is divided into two parts, the first of which discusses a range of technologies that could be used in implementing the architecture and the second of which presents strawman suite of recommendations as a starting point for debate in the telemedicine community. This latter section also notes areas where recommendations should be established but which are not addressed in the strawman suite of recommendations.

It should be noted that the list of options that follows is not complete and is simply meant as a starting point for discussion within the telemedicine community regarding which options are available for the kinds of purposes identified in this document. Likewise, the recommendation of specific products from this list of options does not imply endorsement or a certification of their fitness for use in telemedicine environments and is only meant to foster debate within the telemedicine community.

### 3.3.1 Candidate Technologies and Standards

Because of the multidimensional nature of the architecture (i.e., we're not talking just about medical devices or just about patient records but a broad range of functions), no one technology will meet the needs of every part of the overall architecture. Instead, to answer our question regarding which options are available to us, we need to divide the

problem into smaller problems (i.e, "What options do we have for *this* part of the architecture?").

#### *3.3.1.1 Distribution of Station Components*

In order to allow a station be implemented as a distributed collection of plug-and-play components developed by independent vendors, the mechanism used for implementing the station's core protocols (i.e., registration, leasing, and subscription) need to be standardized. Among the range of options available for the latter need this are:

- CORBA
- The Java Family (specifically, Java Beans, Jini, and Enterprise Java)
- .NET
- Generic Web Services
- Universal Plug-and-Play
- Salutation

##### 3.3.1.1.1 The Common Object Request Broker Architecture

CORBA is the Object Management Group's answer to the question of how to support distributed software solutions that span a variety of platforms. Using a set of interfaces that serve as proxies for remote software components, an "Object Request Broker" that handles the details of how to translate local interactions with the proxies into messages to which the associated remote components can respond and then conveying these responses back to the local components that are interacting with the proxies, and a variety of "services", a programmer can create software systems that deliver the kinds of distributed, plug-and-play capabilities that we have described.

Services of use in implementing the architecture described in this document include the "naming service" and "trader service" make it possible to deliver the kinds of "find by name" and "find by attributes" features that we want in the registry and the "event service" that facilitates the "publish and subscribe" model of component interactions that we have described. In addition, using CORBA's "lifecycle service", CORBA applications have the ability to instantiate objects on remote platforms. Similarly, with CORBA's "externalization service", objects that have been "serialized" can be copied to remote platforms and reinstantiated. Both approaches offer a potential basis for the dynamic configuration capabilities described in our architecture.

##### 3.3.1.1.2 The Java Family

Beginning as a approach for "write once, run anywhere" programming, the world of Java has grow to include the capabilities needed for implementing flexible software solutions in a range of application environments. While representing only a small portion of the Java family, three members of the family are described here because of the close match between their features and those described in this architecture. They are Java Beans, Jini, and Enterprise Java.

Java Beans are the Java world's answer to "components" (i.e., software building blocks that enable the creation of software applications through the knitting together of robust functional blocks). Java Beans support "introspection", which is foundational to a

component's ability to describe itself to others. As building blocks for the kinds of systems that we are discussing, the Java Beans "event" model allows one or more Beans to dynamically subscribe to a specific event message published by a given Bean. Finally, the mechanisms that exist to allow Beans to be retrieved from remote sites on a network and then executed locally are well suited to the dynamic configuration capabilities that we hope to include in our stations.

Jini is an architectural approach that couches everything in a cyber world as a "service". Jini provides programming interfaces that enable components offering these services to advertise their services and that enable other services to lease these services. Built on Java's Remote Method Invocation (RMI) approach to object distribution, Jini allows a component offering a service to present this service as a Java object that can be dynamically loaded onto a client component in order to proxy interactions between this client and the service that the object represents. As this proxying hides the details of the interactions between the interface object and its service, the Jini architecture supports the use of proprietary protocols (a nice feature in our world where it may be a long time before we see architecture-compliant devices that can be plugged into our systems). As with Universal Plug and Play (described below), Jini presumes the presence of a capable IP-based network.

Enterprise Java ("J2EE") exists, as its name implies, to service the needs typical of large organizations. In this sense, it is Java's approach "N-tier" architecture where "legacy" applications and databases are linked to user interface components through layers of generic middleware and enterprise-specific "business logic". Included in this architecture are approaches for integrating user presentation and logic elements of a system and, as such, may provide a reasonable foundation for addressing needs in our architecture like the ability to configure a station's user interface on a user-by-user basis, the ability to present processes in standardized ways (e.g., through "enterprise Java Beans"), and the ability to dynamically map protocols as new user and patent contexts are established. In addition, use of J2EE approaches may help address the desire to integrate telemedicine stations with other information systems in a care delivery environment.

#### 3.3.1.1.3 .NET

A direct competitor to Java's J2EE, .NET is Microsoft's new approach to distributed object programming. Found in Windows XP and related products, .NET is also intended to address the need for the kind of "write once, run anywhere" programming popularized by Java, to support the implementation of enterprise-scale systems, and to facilitate the creation of web service-enabled applications. With respect to the first point, .NET incorporates a "Common Language Runtime" that functions much like Virtual Machine in Java but with one twist – it can support programs written in any .NET-compliant language (which is just about everything *except* Java). With the inclusion of Universal Plug and Play (described below) in the .NET strategy, Microsoft's solution offers a mechanism for supporting our architectures "publish and subscribe" approach to component interconnection. .NET's context management capabilities may serve as a reasonable foundation for the contexts contained in our architecture. The web service support looks like a good candidate for implementing many of the station-to-station interactions.

#### 3.3.1.1.4 Generic Web Services

Spurred by the desire to support effective business-to-business e-commerce, the World Wide Web Consortium is pursuing the development of an XML-based set of protocols that allow a entity to discover and employ the services offered by remotely located entities. Using HTTP as their transport mechanism, these web services facilitate both unicast and multicast communication of messages from one entity to one or more recipients. These messages can be used as simply that – messages – or can be used to form more sophisticated mechanisms, such as remote procedure calls.

As things currently stand, there seems to be general agreement among all of the major vendors on the underlying mechanisms used to deliver these services. In particular, basic messages exchanged in web services are encoded using XML and transported using the Simple Object Access Protocol (SOAP). Because SOAP typically runs over HTTP – the central protocol of the World Wide Web – it can generally reach anywhere the web can reach, even through firewalls that might normally block similar kinds of protocols. Web services offered by an organization are advertised in web-based registries and discovered using the suites of services defined by the Universal Description, Discovery, and Integration specification (UDDI). Services published in this way are described using the Web Services Description Language (WSDL) – a relatively flexible mechanism that allows for the definition of both the abstract functions offered by a given service and the protocol mechanisms used to support these functions.

While these components provide a reasonable foundation for vendor-independent delivery of services, by themselves, their capabilities fall short of what is needed by certain applications. To meet this need, efforts like ebXML (a business to business web service approach) are being supported by different business sectors (e.g., ebXML is being actively pursued by the HL7 community).

#### 3.3.1.1.5 Universal Plug-and-Play

A consortium of approximately 200 vendors led by Microsoft is responsible for this specification. The goal of this group's efforts is to create an interoperability approach for net-aware peripherals that is based on a collection of open protocols. The key idea behind this specification is to allow devices on a network to dynamically discover each other's existence and to explore and subscribe to the services that different devices offer. Aimed allowing the formation of ad hoc peer relationships within a network, UPnP offers a lightweight way of connecting devices that join the network.

The UPnP architecture defines three key concepts: devices, control points, and services (Figure 76). Devices provide services and can also contain other devices (e.g., a vital signs monitor might contain several devices as well as offer certain services that apply to the entire monitor, such as "reset" or "self test"). Control points are consumers of the services offered by devices. They do their job by directly controlling a given service, by "subscribing" to "events" that services may publish, or both. Services consist of those atomic actions that a given device will perform. In addition, each service will typically

expose some portion of its internal "state" in such a way that allow control points to monitor changes in the service's state and to respond accordingly. When commands are received from a control point, a service will execute it and return a response. Often, this action will result in a change in the service's internal state and, if any subscribers exist for this sort of change, they will be notified.

As shown in Figure 77, UPnP assumes the presence of an IP network. UPnP devices joining a network start off by trying to establish an IP address for themselves (using the device to learn more about its services and its composition (i.e., whether the device DHCP or "Auto IP"). Once connected at this level, a device will then announce its presence to the network by means of a Simple Service Discovery Protocol (SSDP) broadcast message. Control points interested in the device can then communicate withitself contains any other devices. If it desires, a control point can then send a control message to one of the device's services. This message is formatted in XML and conveyed using the SOAP protocol. In response, the service will return its own SOAP message containing status, faults, etc. As each service also publishes a list of variables that models its internal state, control points can also choose to subscribe to event messages that are published using the General Event Notification Architecture (GENA) protocol (a simple publish and subscribe mechanism).

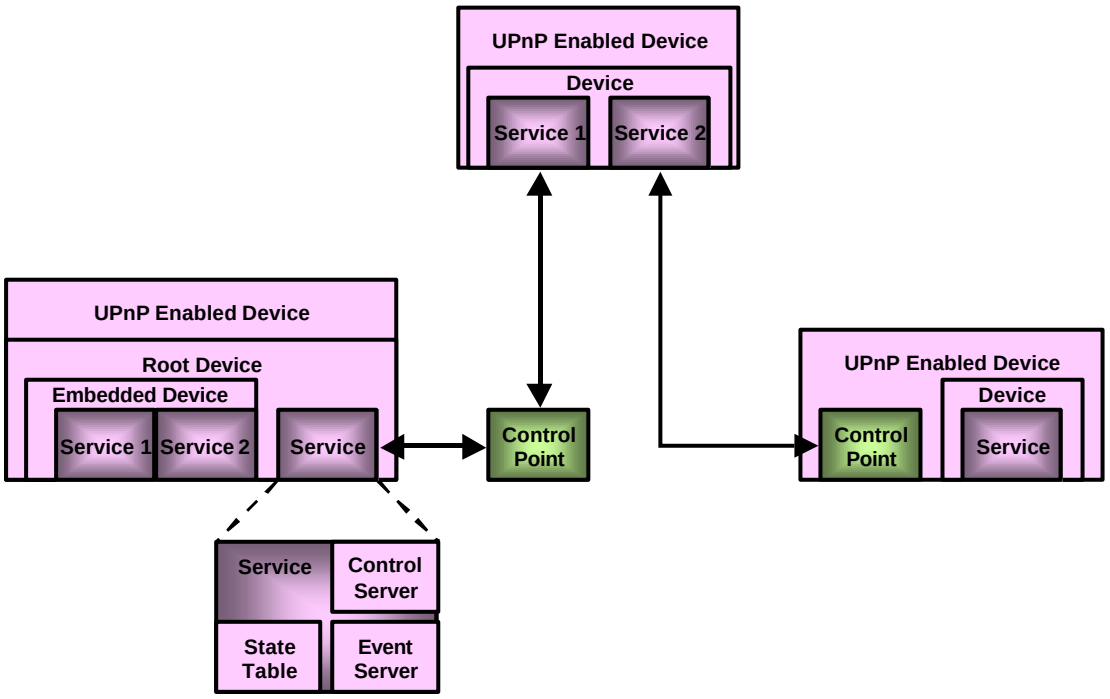


**Figure 76. UPnP Architecture**

Before moving on to Salutation, there are a couple more things worth noting. Everything described so far is true of *every* UPnP device and are realized in the layers marked "UPnP Device Architecture Defined" and below in Figure 77 (i.e., the capabilities in these layers are part of every UPnP device). It is above this point that UPnP devices begin to

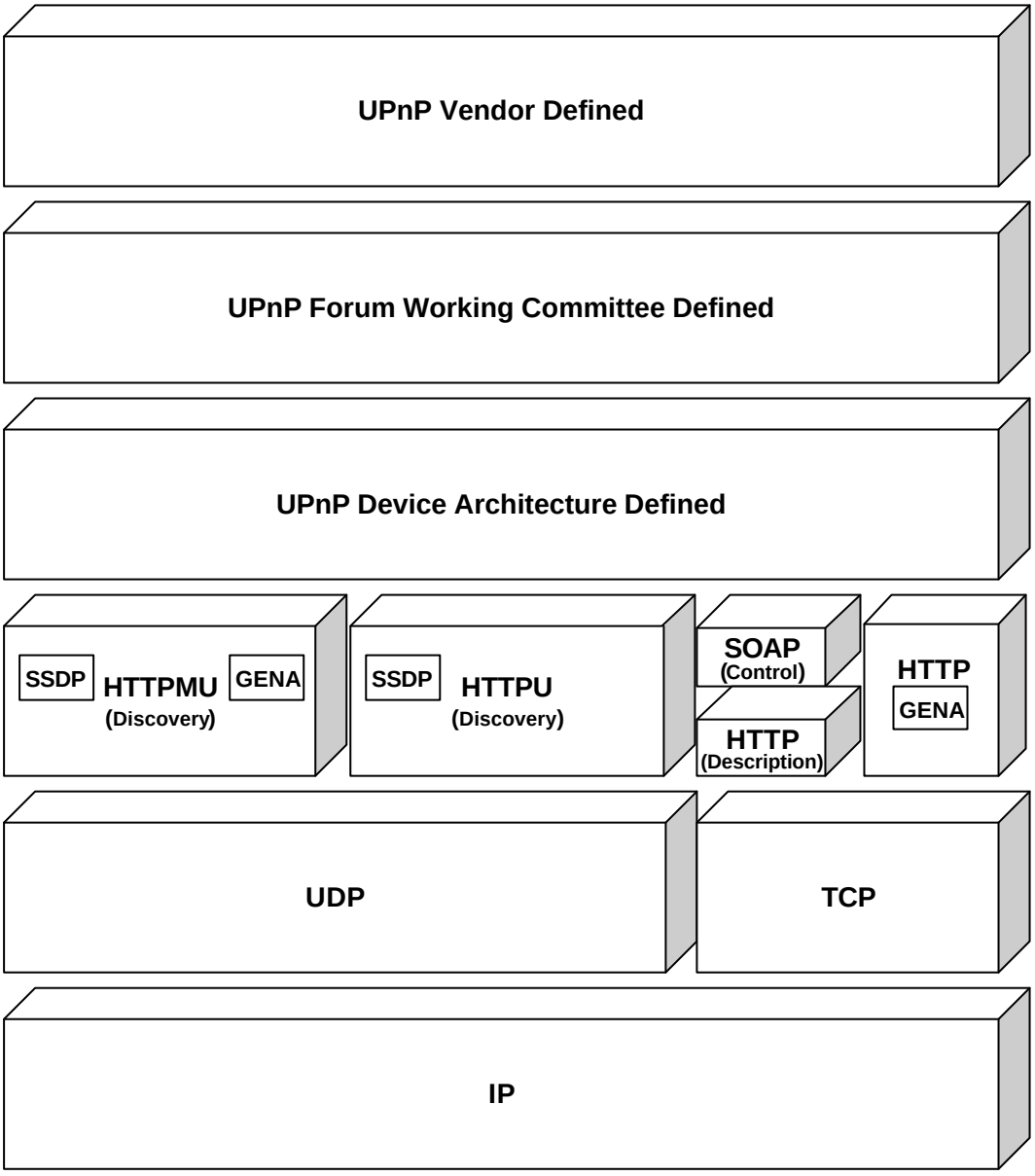


**Figure 77. UPnP Protocol Stack**

differentiate themselves from one another. The second layer down in the stack represents a series of service-specific interfaces that are being standardized by the UPnP Forum.

These interfaces define the minimal set of actions and state variables that any given *kind of service* must expose. This approach ensures that a control point wanting to utilize a particular type of service can do so with any manufacturer's device. At the same time, because the Forum recognizes that vendors may want to add their own features to a given kind of device, the protocol stack includes a top layer that accommodates vendor specific extensions to a device. By implication, the vendor would also need to supply a control

point (or a way of extending "standard" control points) to accommodate these extra features.

Finally, even though the UPnP specification has remained relatively stable and has garnered a certain degree of support, a modified specification should soon be released that moves UPnP more towards the web services model and that adds certain features, such as authentication and encryption, that had been lacking in the original specification.

#### 3.3.1.1.6 Salutation

Created by a consortium formed in the mid '90s, Salutation is intended to address the problem of how to allow devices to dynamically associate with other components (e.g., how to let a PDA discover and employ a printer). Using a "Salutation Manager", a compliant device is able to advertise its services, discover the services offered by other devices, determine the availability of a service offered by another device, and establish sessions with other devices in which messages flow back and forth between devices. All interactions of this involving Salutation Manager to Salutation Manager communications are based on Sun's Remote Procedure Call mechanism. Communications between servers and clients in this architecture can be done completely "native" with the Salutation Manager being used only to establish session connections, can be Salutation-based with every aspect of the communications being provided by a Salutation-specified approach, or can be accomplished using some combination of the two approaches. Salutation does not specify a particular transport mechanism and, as such, defines the Transport Interface as a means of keeping the Salutation Manager and servers and clients transport neutral.

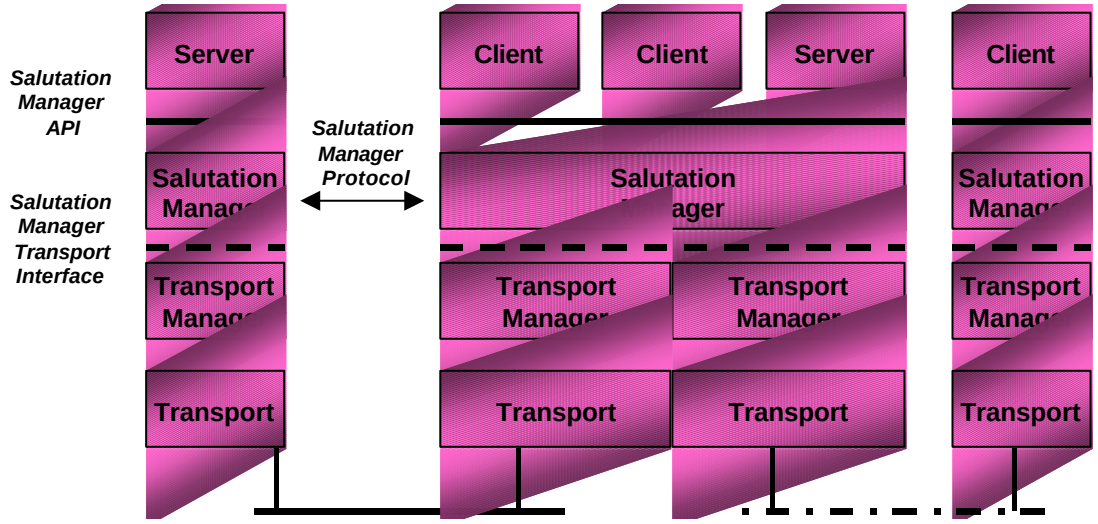


**Figure 78. The Salutation Architecture**

### 3.3.1.2 Internal Communications Bus

In addition to standardizing the approach to implementing the architecture's core protocols, the mechanisms for implementing the internal communications bus must also

be standardized if the plug-and-play operation of distributed system components is to be achieved. Technologies and standards of use in this section of the architecture include:

- IP-based home networks and LANs with QOS protocols
- Firewire
- HAVI

#### 3.3.1.2.1 IP-Based Local Area Networks

Whereas not too many years ago it looked as though IP-based networking might soon be replaced by other "more capable" approaches, the penetration of 100 megabit Ethernet over the last few years and the appearance of off-the-shelf gigabit solutions bode well for IP networking's staying power. In addition, in the wireless and home networking markets, Ethernet solutions for computer connectivity seem to being prevailing (with 802.11b seeming to win in the wireless sector and HomePNA doing well in the homes). As noted in the discussions above, a number of the distributed object schemes assume the presence of IP infrastructures. Given these things, IP-based networking has to be considered a contender for the kinds of station-internal communications defined in our architecture.

#### 3.3.1.2.2 Firewire

Originally intended as a serial bus for peer-to-peer interconnection of devices located in close proximity to one another, Firewire (a.k.a., IEEE 1394 or i.Link) was been strongly embraced by the audio-visual device community, largely because of its ability to deliver high bandwidth, on-demand connectivity between devices. Unlike IP-based systems that employ "best effort" approaches to delivering data between devices on a network, Firewire's "isochronous" channels provide guaranteed data rates for those applications that need them.

Because of distance limitations in Firewire's original twisted pair cabling, Firewire was never seen as a viable medium for local area networking; however, the most recently released version of the IEEE 1394 specification accommodates fiber optic communications (note to mentioned gigabit data rates) which significantly extend the reach of this communications medium. When combined with the fact that IP networking over Firewire has already been demonstrated, Firewire is also a strong candidate for our architecture's internal communications bus.

#### 3.3.1.2.3 Home Audio Video Interconnect

While it is an architecture rather than a communications medium, Home Audio Video Interconnect (HAVi) is included in this section because the most important aspect of this architecture is how it handles communications (Figure 79). Developed cooperatively by eight of the world's leading consumer electronics manufacturers, this specification's goal is to allow for true plug and play operation of home entertainment products. Devices on a HAVi network are able to readily share resources with one another (e.g., a television in one room can host a user interface used to control a DVD player located in another room). Because of its ability to deliver high bandwidth, isochronous communications, Firewire is the underlying data transport medium employed in HAVi.

As shown in the figure, HAVi consists largely of a set of software elements that bridge the gap between custom software applications and vendor specific platforms intended to host the applications. The 1394 Communications Media Manager exists to provide the raw asynchronous and isochronous communication services on which HAVi is built. The Messaging System manages the asynchronous messaging that occurs between HAVi components. The Registy in HAVi serves essentially the same function as the one described in our architecture. Using the Messaging System, the Event Manager supports the passing of events between components. The Stream Manager handles real-time data connectivity for those devices that require continuously flowing data. As currently defined, this mechanism allows for "typing" of streams that allows things like required bandwidth and data encoding to be standardized for different kinds of isochronous connections. The Resource Manager exists to handle resource contentions and scheduling of future resource utilization. The DCM Manager handles the adding and removing (much like the proxies in our architecture) of Device Control Modules (DCMs), the software components responsible for presenting each device's APIs to all

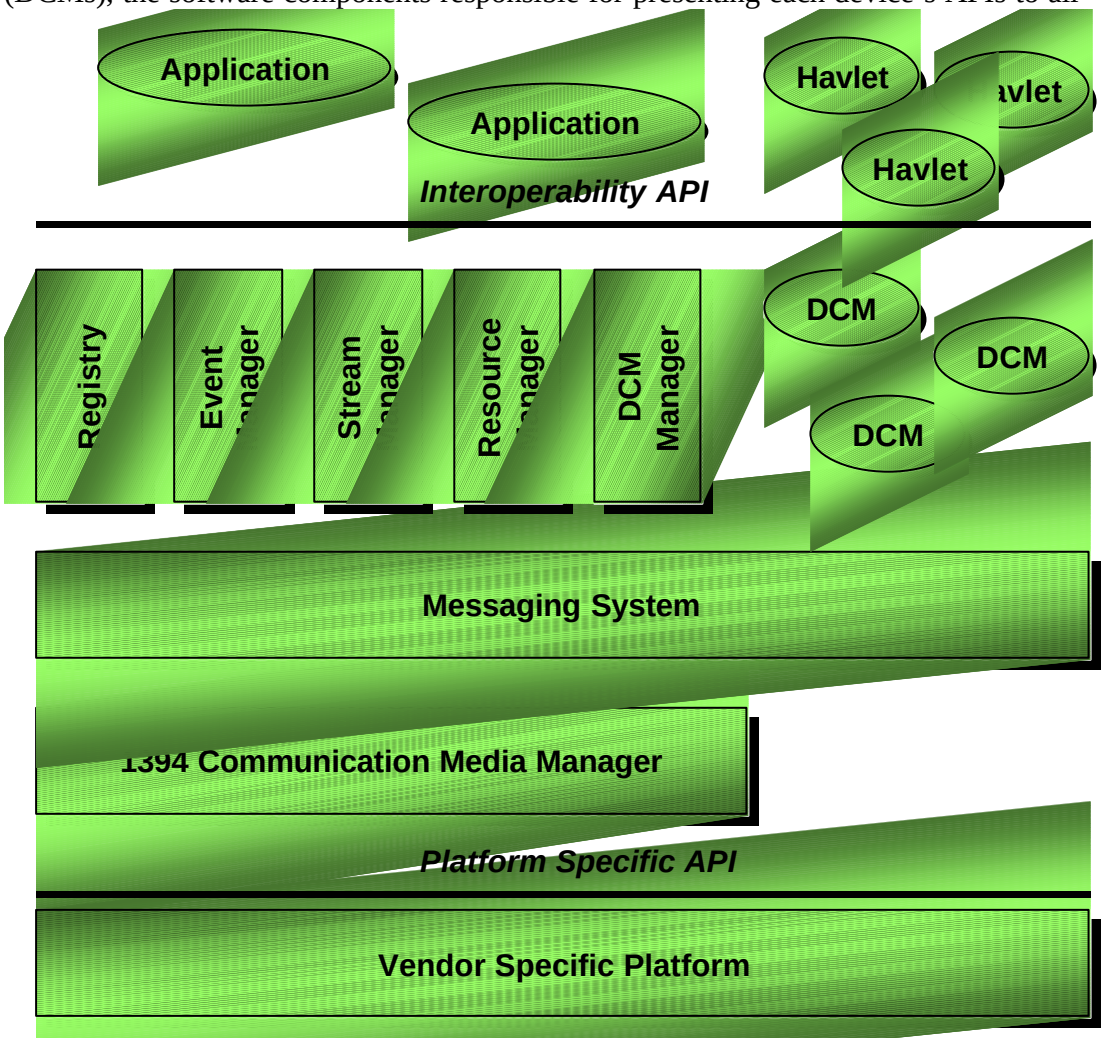


**Figure 79. The HAVi Architecture**

other devices on the network. Like "services" in Jini, Functional Control Modules (FCMs) contained within DCMs provide interfaces for each controllable function within a device. While DCMs typically front-end home entertainment devices, the architecture supports the notion of DCMs acting as bridges to services (such as the Web) that are external to the HAVi network. Havlets are Java applications that can be used to provide user interfaces for various devices. Applications are platform-independent software components that make use of the various services provided in a HAVi network to provide various services wanted by end users.

Finally, the HAVi architecture supports the uploading of HAVi "code units" to a HAVi network. Code units are Java JAR files that are packaged with several other security oriented files used to established the code unit's authenticity

#### *3.3.1.3 Device Buses*

As noted earlier, device buses exist in the architecture as a means of supporting the low-level connectivity functions needed in order to support plug and play operation of devices. Device bus candidates include:

- IrDA
- Bluetooth
- USB
- Firewire
- The PCMCIA Family

##### 3.3.1.3.1 IrDA

The IrDa specification, published by the Infrared Data Association, governs the infrared communications technology used in most laptops and PDAs. Capable of transfer rates of up to 16 megabits/second but generally used at much lower data rates, this standard is typically used in environments where low power wireless connections are a plus. The IrDA family of specifications allows for IrDA's use both as a cable replacement (i.e., in point-to-point mode) and in IP-networking mode.

##### 3.3.1.3.2 Bluetooth

An emerging wireless alternative to IrDA, this RF-based communications standard enjoys the support of a very large industry group. As with IrDA, Bluetooth can be used in both point to point and multipoint modes. Slow time to market and some compatibility issues with 802.11b networking devices have caused some people to question Bluetooth's long term viability, especially in light of the fact that the order of magnitude price differentials in favor of Bluetooth over 802.11b devices have yet to materialize.

##### 3.3.1.3.3 USB

Designed as a replacement for simple serial interfaces on the personal computer, the Universal Serial Bus (USB) is much like IEEE 1394 (Firewire) in its operation. USB allows multiple low-speed devices to be attached to the bus in plug-and-play fashion and for buses to be chained together to allow a relatively large number of devices to be attached to a PC via a single connector on the PC. Originally designed to run at 12 megabits per second, the most recent revision of the specification bumps the speed into

the 100s of megabits range – fast enough to support video devices. As with Firewire, USB supports isochronous communications, a feature that is key to applications that demand a guaranteed data transfer rate. Unlike Firewire that allows "equality" among all nodes, USB differentiates between the host platform and the rest of the nodes that exist on the bus.

#### 3.3.1.3.4 Firewire

While recent changes to Firewire have positioned it for local area networking in certain application environments, Firewire's first calling was as a high speed serial bus. More capable than USB, it is also less prevalent, with the latter being incorporated in almost all new PCs.

#### 3.3.1.3.5 The PCMCIA Family

Originated in the early '90s as means of extending the capabilities of portables, PC cards have moved from 8 and 16 bit implementations used at a time when the ISA bus was the document internal bus in PCs to the 32 bit "CardBus" standard designed for use with PCs built around the PCI bus. Now the Personal Computer Memory Card Industry Association (PCMCIA) is in the process of fielding a new standard called "CardBay" that addresses the needs of modern mobile computing devices. By marrying a physical card socket format that maintains compatibility with existing PC card technologies with a USB interface between this socket and the PC's internal components, CardBay provides add-in cards with a high-speed interconnect to the host platform. With a power source of card internal operations and card detect and query mechanisms, CardBay may be well suited to supporting plug-and-play operation of certain power-intensive device classes.

### *3.3.1.4 External Communications Media*

Whether the telemedicine station is composed of a group of distributed components interconnected by a local network or consists of a monolithic base station that hosts integrated or detachable peripherals, the station will present a set of external interfaces to the outside world. Given this, the telemedicine community must decide which wide-area communication methods it needs to support. Depending on the utilization concept, a number of possibilities exist. Among these are:

- Analog phone
- ISDN
- Traditional Ethernets
- Wireless

#### 3.3.1.4.1 Analog Phone

Often referred to as "POTS" ("Plain Old Telephone System"), the strength of this approach is in the pervasiveness of the medium. Compared to digital cable that has penetrated just under 25% of the U.S. consumer market and its accompanying cable modems that are used in a little under 9% of U.S. homes) and the telephone companies' broadband offering, DSL, which exists in a little under 4% of these homes, phone connectivity is pervasive, being found in 96% of U.S. homes. Internationally, the total number of wired phones is somewhere around 1 billion. While currently exceeding most

current wireless capabilities, analog phones provide relatively low bandwidths (nominally 56 kilobits/second per International Telecommunications Union standards).

#### 3.3.1.4.2 ISDN

Introduced a number of years ago, the Integrated Services Digital Network phone system provides higher bandwidths that POTS, with 128 kilobits/second available in the Basic Rate Interface and 1.544 megabits/second in the Primary Rate Interface (more commonly found in business environments than in the home). As with DSL, ISDN support simultaneous voice and data services. Techniques also exist in the ISDN world for using multiple phone lines to deliver additional increments of bandwidth.

#### 3.3.1.4.3 Traditional Ethernets

In campus- or building-internal environments in which all stations in the system share a common infrastructure, traditional twisted pair (10BaseT/100BaseT) Ethernet is a good candidate for interconnection of stations. The same can be said of wireless Ethernets (802.11b or 802.11a) designed to support local area networking. In home environments, cable modems, digital subscriber line (DSL) modems, and other home gateways that support a variety of external communications also provide support for Internet access. Current DLS offerings typically run in the 100's of kilobits/second range for symmetric operations and can reach up into the of megabits/second for asymmetric depending on the flavor of DSL and a variety of physical factors, such as the distance between the DSL modem and the supporting equipment at the phone company's switching center. Cable modems typically exceed this figure.

#### 3.3.1.4.4 Wireless

There appear to be a number of key developments in this area that are all extensions of Global Services for Mobile Communications (GSM) – the dominant approach to voice and data communications in the mobile world. Included in this are General Packet Radio Service (GPRS) – an enhancement to GSM that allows for persistent Internet connectivity and faster data communications – and Enhanced Data for GSM Evolution (EDGE) – a body of work focused on extending current GSM network to enable them to deliver high bandwidth content (e.g., video delivered at speeds up to 384 kilobits/second). 3GSM – an umbrella effort by the 3rd Generation Partnership Program (3GPP) that will support third generation mobile multimedia services – will ultimately replace these intermediate measures with a new of set of capabilities offering mutli-megabit video, audio, and data networking.

### *3.3.1.5 User Interfaces*

A number of options are available for controlling telemedicine station functions and for extracting information from these devices. These include:

- a common look and feel and shared context
- a common, custom framework
- browser-based interfaces (to include WML apps)
- voice I/O systems
- integrated controls and displays

#### 3.3.1.5.1 A Common Look and Feel and Shared Interface Context

Rather than trying to integrate all station user interface capabilities into a common software-based interface, the telemedicine community could decide to create suites of stand-alone applications that can be launched and terminated independently of one another and yet operate in a coordinated fashion through the use of a shared interface context. For example, the patient record system available to a station could be viewed and controlled by one applications while each of the devices in the station are controlled by their own or by applications that implement sophisticated multi-device protocols. As key context elements change on the station (e.g., which patient is being serviced or which user is operating the station), the various user interface applications would be notified so that they could respond in a unified fashion to these changes.

To contribute to the sense that all of these individual applications are part of the same station, the telemedicine community could establish rules regarding the look and feel of compliant applications. This would control how interface screens are organized, rules regarding when to use various elements of the screen, restrictions on which terms and icons to use, etc. If used, there are a number of style-guides of this sort available today that could be adapted for use by the community.

#### 3.3.1.5.2 Common, Custom Framework

A second software-based approach is to create a common graphical interface based on standards interface technology (e.g., the Windows interface) that can be extended through the use of add-ons. In this context, the custom user interface "applets" used to deliver display and control for each station capability would install themselves in the framework through some sort of interface registration process. This process would cause the appropriate controls to be installed in the framework and would establish the rules for when these controls are to be presented to users and what to do when the controls are triggered. The applets launched by these controls would then operate within the context of the common framework, either using resources (e.g., screen real estate) provided by the framework or operating as "child windows" to the framework.

#### 3.3.1.5.3 Browser-based Interfaces

Like the previous approach, browser-based interfaces provide a common framework in which a number of applications can operate as one. Unlike the custom framework, this approach has the advantage of drawing on the power of a large body already available in the Internet world. Whereas the previous two approaches assume that the standardized user interface application interfaces (i.e., those service interfaces that these components present to the rest of a telemedicine station) are bound to the applications or applets running on interface computer, the browser-based approach also allows for these interfaces to be run on "server" side of a system and user interfaces to be implemented as completely standard (e.g., HTML or WML) web interfaces.

#### 3.3.1.5.4 Voice I/O Systems

While still somewhat immature, voice technology is emerging as a useful tool in certain application settings. Beyond simply supplementing the keyboard (i.e., entering a command or data verbally rather than typing it in), technologies are emerging that promise to allow wholesale replacement of traditional user interface mechanisms. One

such technology is VoiceXML, a markup language for voice I/O applications (www.vxml.org). Related World Wide Web Consortium efforts in this area point to a solid future for "voice browsing".

#### 3.3.1.5.5 Integrated Controls and Displays

In some situations, it will be desirable for interface controls and displays to be integrated directly into various station components. For instance, in medical instruments, controls (e.g., start/stop buttons and status lights) that allow their users to operate independently from the station's centralized user interface support smoother, more natural interactions.

### *3.3.1.6 Medical Devices*

In the case of medical devices, the choice will not be which approach is *the* approach but which is right for a given need. Options to consider include:

- IEEE 1073
- POCT
- DICOM (note relationship to certain non-imaging devices)
- TWAIN and EXIF

#### 3.3.1.6.1 IEEE 1073

Known as the "Medical Information Bus" or "MIB", the historical focus of this specification was on plug-and-play operation of devices common to the bedside in hospital environments (e.g., vital signs monitors, infusion pumps, defibrillators). The IEEE 1073 family of specifications standardizes the logical models for various medical devices, the application level protocols needed to allow these devices to talk with "bedside controllers", and the lower level protocols used to transport these application level messages. While early implementations of the standard relied on proprietary connectors and transport mechanisms, recent years have seen 1073 begin to embrace the use of industry standard lower level protocols. In addition, whereas 1073's historical focus has been on bedside monitoring in hospital environment, members of the 1073 committee have expressed a growing interest in additional utilization scenarios, including those related to the needs of the telemedicine world.

#### 3.3.1.6.2 POCT

The Point Of Care Test (POCT) specification is the product of the Point Of Care Communications Interoperability Consortium (POCCIC), a short-term effort by a coalition of manufacturers point of care analytical devices. Drawing on the lower layers work of the 1073 committee and upper layer concepts from HL7, the POCCIC drafted a specification that allows point of care devices to connect in wired or wireless fashion to a network and to interoperate with remote "data manager" software processes, irrespective of where these data managers may be located on the network. Given the POCT focus on lab functionality, the list of devices addressed by this group complements those handled by the 1073 committee.

#### 3.3.1.6.3 DICOM

As the premier medical imaging interoperability specification, the Digital Imaging and Communications in Medicine standard defines the information storage formats and

protocols needed for handling of digital clinical imagery. As such, the standard provides for computer-to-computer and computer-to-device (e.g., various acquisition systems or printers) communications across a broad range of imaging modalities. Beyond pure imaging applications, DICOM also provides for the management of certain other types of data, such as waveforms and the contextual information needed to annotate individual images or sets of images.

#### 3.3.1.6.4 TWAIN and PTP

While designed for use in the PC environment, digital cameras and scanners have found their way into common use in the telemedicine community where they enable local caregivers to capture clinical images. These devices can employ a variety of physical media (including wired, wireless, and storage device). Depending on the nature of the device, it may support the TWAIN ("Tool Without An Important Name") interface or, in the case of digital cameras, the "Picture Transfer Protocol". The TWAIN specification is the older of the two. It specifies both a three-layer architecture that allows a compliant software application to access the services of one or more raster devices without having to know the unique details of each such device. By comparison, PTP focuses primarily on the communication of images from and to "digital still photography devices" (DSPDs) and, secondarily, on allowing external devices to control the operation of a DSPD (e.g., to configure a camera or to tell it to snap a picture). Designed to be independent of the underlying data transport mechanism, PTP can be made to run over a variety of media. The companion EXIF and TIFF/EP file format specifications (that also handle audio recordings generated by cameras) and the Design Rule for Camera File System (DCF) specification that standardizes on-camera file system structure and behavior round out this specification's family.

### *3.3.1.7 Patient Record Repository*

In specifying how to implement the repository itself, several on-going efforts are potentials candidates. These include:

- The Good European Health Record (GEHR),
- HL7's Clinical Document Architecture (CDA),
- OMG's Clinical Observation Access Service (COAS) and Clinical Image Access Service (CIAS) and their supporting specifications, and
- CEN ENV 13606

#### 3.3.1.7.1 GEHR

Originally developed in Europe, work to extend this patient record architecture is now underway in Australia as well. Based on two key elements – a "reference object model" (ROM) and an "archetypes model" – GEHR strives to overcome many of the problems introduced by approaches that attempt to create the "mother of all medical records". Where as the ROM provides a limited set of building blocks from which many things can be constructed, the archetypes constrain the aggregation of these building blocks to meaningful constructs. Given this, GEHR appears to provide ways for disparate electronic medical record systems to interact (e.g., GEHR information talks to interacting with each of the other repository approaches discussed in this section).

#### 3.3.1.7.2 HL7 CDA

HL7's Clinical Document Architecture (CDA) is a markup standard for use in exchanging clinical document. As such it specifies both the structure and semantics of these documents. CDA has been accepted by ANSI and is likely to gain in stature over time. As part of HL7's move toward XML, one of the goals for this specification was the creation of documents that can be both read by humans and parsed by computers. The plans for CDA include the ability to support three levels of markup

#### 3.3.1.7.3 COAS/CIAS

COAS is the OMG's Healthcare Domain Task Force (HDTF) approach to storing and retrieving clinical data. Inspired by the original GEHR work, the COAS specification defines a structure for storing and retrieving clinical observations that is independent from the specific content being stored using these mechanisms. While this structure may be the actual structure used by the patient record software, COAS may also be nothing more than a "wrapper" used on a legacy record structure that provides the legacy structure with a standard way of presenting its contents to the outside world.

CIAS is the HDTF analog to COAS that is used for retrieving and managing clinical imagery for servers and for translating certain formats (such as DICOM's) into Internet-standard image formats. Not meant to be a replacement to DICOM, CIAS provides a means of exchanging images when the in environments where the full capabilities of DICOM are not required. The CIAS specification also provides a means of encapsulating both DICOM and non-DICOM image stores behind a uniform service interface such that clients of a server need know only one image retrieval mechanism.

#### 3.3.1.7.4 CEN ENV 13606

A European standard for health records, ENV13606 define the content and structure of electronic health records that are communicated between institutions as well as means of specifying access control rules for these records and protocols for exchanging these records. Much of its inspiration was drawn from the early GEHR work

### *3.3.1.8 Processing and Protocols*

The station's intelligence, as embodied in its processing and protocol components, can take the form of either hardware or software and, in the case of the latter, may be compiled or interpreted modules. In both cases, a component of this sort may be collocated with the rest of the station it supports (e.g., a piece of software purchased by its users) or may be remotely located (e.g., a web-based service that front-ends specialized capabilities leased from a company on an as-needed or subscription basis). In specifying the form of software modules, the telemedicine community might take any of three approaches:

- allow for arbitrary platform
- specify a single approved platform
- support a suite of common platforms

#### 3.3.1.8.1 Allow For Arbitrary Platform

In this approach, the community decides that all that matters is the interface mechanism used to present the services that these components provide to the rest of components in a station. In this approach, vendors can target specific common platforms (e.g., Windows, Linux, or a Java Virtual Machine running on any number of underlying platforms) that support their business strategy or can even develop components that are installed on custom platforms.

#### 3.3.1.8.2 Specify A Single Approved Platform

At the other end of the spectrum, the community could choose a single platform, as the HAVi community has done with the Java modules that run on JVMs embedded in HAVi devices. The advantage of this approach is that any vendor's processing components will work on any other vendor's platforms.

#### 3.3.1.8.3 Support A Suite Of Common Platforms

In this third approach, the telemedicine community would agree to the suite of platforms that they want to support and vendors would then offer, for each of the pluggable processing components that they produced, versions that run on each of the platforms in the suite. Some technologies supporting this option include:

### *3.3.1.9 Contexts*

While Microsoft's .Net My Services building blocks with their Passport security services used to be the only game in town, recent months have seen the emergence of the Liberty Alliance as a potential contender for maintenance of net-based user context information. On a localized level, the CCOW work within HL7 provides some of the capabilities needed for contexts in telemedicine systems.

#### 3.3.1.9.1 .Net My Services

An on-line, XML-based approach to centralizing information about individuals, My Services (formerly called "Hailstorm") are a central pillar in Microsoft's vision of net-centric computing. By storing their information in a single place on the Internet, any application needing access to this sort of information can retrieve it (subject to the person's approval), thereby allowing the person to avoid repeatedly entering the same information into different services. The list of initial services proposed by Microsoft include:

- **.NET Profile.** Name, nickname, special dates, picture, address.
- **.NET Contacts.** Electronic relationships/address book.
- **.NET Locations.** Electronic and geographical location and rendezvous.
- **.NET Alerts.** Alert subscription, management, and routing.
- **.NET Presence.** Online, offline, busy, free, which device(s) to send alerts to.
- **.NET Inbox.** Inbox items like e-mail and voice mail, including existing mail systems.
- **.NET Calendar.** Time and task management.
- **.NET Documents.** Raw document storage.
- **.NET ApplicationSettings.** Application settings.
- **.NET FavoriteWebSites.** Favorite URLs and other Web identifiers.

- **.NET Wallet.** Receipts, payment instruments, coupons, and other transaction records.
- **.NET Devices.** Device settings, capabilities.
- **.NET services.** Services provided for an identity.
- **.NET Lists.** General purpose lists.
- **.NET Categories.** A way to group lists.

#### 3.3.1.9.2 Liberty Alliance

Initially snubbed by Microsoft, the Liberty Alliance Project has gained significant momentum in recent months with the addition of key players, such as Mastercard, Visa, and AOL, to their membership roles. While still immature (specifications are not publicly available whereas Microsoft's product is up and running today), this "federated identity" alternative to Microsoft's "store it all in one place" appears likely to emerge victorious now that Microsoft has begun to make noises about ensuring the ability of its My Services offerings to interact with the consortium's products.

#### 3.3.1.9.3 CCOW

Unlike My Services and the Liberty Alliance, where the focus is on establishing a user's identity on-line, the Clinical Context Object Workgroup (CCOW) specification allows multiple applications from different vendors to share a common context (e.g., which patient is currently being handled by the caregiver or which encounter is being addressed). In as much as CCOW has been adopted by HL7, is being incorporated into many major HIS applications, and allows a user to carry their "work space" with them, the specification should be considered as a candidate.

### *3.3.1.10 External Communications Management*

To support the dynamic communications environment described in this document, mechanisms are needed to manage all available bandwidth as a pool, to monitor and reconfigure channels (addresses, protocols, etc.) on the fly, and to ensure certain quality of service guarantees. Both the IETF and ITU have on-going work in this area and, as noted earlier in this paper, much of this quite nascent. Given this, the author chooses to leave suggestions to readers who care to advance a position stating why H.245 or RSVP or some other approach should be considered.

### *3.3.1.11 Session Management*

In setting up station-to-station interactions, there is a need for an client station to inquire about the capabilities of the server station. This will include, once each other's identities have been authenticated, allowing for discovery of services offered, and then leasing of these services. Because this information is intimately associated the overall architectural approach being advocated in this document, the protocol used to support this capability will be specific to whatever interoperability architecture is eventually chosen by the telemedicine community. Even so, the community can draw on existing work as a foundation for development of this protocol. Among others, candidates can include the containment tree work of ISO (used today by IEEE 1073 and XML document technologies.

### *3.3.1.12 Patient Record Communications*

For each of the patient record repository approaches described above, the owning organizations also define mechanisms for allowing data stored in these repositories to be shared between using entities.

#### 3.3.1.12.1 HL7 Approach

As HL7 will tell you, the "7" in the name stands for "layer 7" in the ISO network protocol stack. This the "application layer protocol" used by two applications to communicate with one another independent of the underlying means used to communicate the data contained in this protocol. Not surprisingly then, most of the work done to date in HL7 has focused on how to communicate healthcare related data between healthcare enclaves (e.g., hospital to hospital or hospital to payer). While the existing approved standard, HL7 version 2.3, specifies a set of text-based, delimited strings used for accomplishing various data exchanges, a tremendous amount of work on version 3.0 has focused on the development of formalized models that can be automatically and unambiguously translated into a variety of message formats.

#### 3.3.1.12.2 OMG Approach

Both COAS and CIAS support the notion of distributed storage of medical records. To fully exploit their power, the OMG HDTF has defined a number of other companion capabilities. These include PIDS, TQS, HILS, and RAD.

The Patient Identification Service (PIDS) addresses the problem of how to handle the variety of identification schemes that are likely to be used across disparate patient record systems. In queries across widely distributed, yet federated, systems, PIDS handles the problem of resolving identifiers such that the querying entity need know only one identifier for each patient of interest in the query.

The Terminology Query Server (TQS) exists to translate between the various coding schemes used in different patient record systems. As with PIDS, TQS allows a querying entity to formulate queries in its own "native tongue" and then translates these queries and the associated responses as needed to accommodate the needs of each platform involved in supporting the query.

The Heterogeneous Information Locator Service (HILS) allows a querying entity to determine the location of all repositories that might contain information of interest to a given query.

The Resource Access Decision (RAD) specification defines a flexible, object-oriented approach to implementing access control policies for healthcare information.

#### 3.3.1.12.3 GEHR Approach

As described earlier, GEHR's primary function is to provide for communication of data between potentially different types of record stores. As such, the GEHR kernel (a software entity being developed in the Australian GEHR work in order to prove out the specification) provides for the ability to translate between various archetypes representing

messages formats from different exchange protocols and for translation in and out of native storage formats.

### *3.3.1.13 Imagery Communications*

In as much as telemedicine stations will be able to store still and moving imagery, means for discovering the existence of these files on remote stations and transferring them between stations are required. A few of the options for this include:

- DICOM
- CIAS
- Other

#### 3.3.1.13.1 DICOM

As the primary interoperability specification in the digital radiology community, the Digital Imaging and Communications in Medicine (DICOM) standard is an expansive specification that defines the structure of data to be stored in a variety of radiology applications along with the messages used to convey this information between various components of a digital radiology network. Much emphasis in the standard is placed on the conformance mechanisms needed to ensure users that different components are capable of interoperation. The standard allows for communication to radiology data over standard IP network, over point-to-point links, and via removable media.

#### 3.3.1.13.2 CIAS

As noted above, CIAS was designed to allow for standardized communication of clinical images of various types, including those generated by DICOM systems.

#### 3.3.1.13.3 Other

While DICOM and CIAS provide a clinical "context" for exchange of clinical images, it is entirely possible to use other file transfer mechanisms (such as FTP described below) to enable searching and downloading of files from "folders" on remote machines.

### *3.3.1.14 Videoconferencing*

In the videoconferencing world, a number of standards have been established corresponding to different media over which the conference is to be conducted or to the organizations responsible for their generations. Among these are:

- H.323
- H.324
- SIP

#### 3.3.1.14.1 H.323

Developed by the International Telecommunications Union (ITU) as the standard for packet-based multimedia communications, H.323 provides means for conducting audio, video, and data-conferencing over networks in which quality of service is not guaranteed by the network. Based on a suite of other specifications, H.323 enables both point-to-point and multi-point conferencing. For conferencing over IP networks, it is currently the most mature of the competing standards and has been implemented by a wide range of vendors. As an open standard, it allows for vendor independence in construction of

conferencing networks. Through the use of H.323-compliant gateways, it allows for integration with other kinds of conferencing "terminals", such as ISDN-based videophones or simple analog telephones (i.e., POTS). H.323 has built-in quality of service mechanisms that enable bandwidth within a network to be managed both by limiting the number of calls running at any given point in time on the network and by limiting the amount of bandwidth available to any given call.

#### 3.3.1.14.2 H.324

Another ITU specification, H.324 addresses conferencing over analog phones networks.

#### 3.3.1.14.3 SIP

While lagging behind H.323 in terms of introduction, the Internet Engineering Task Force's Session Initiation Protocol (SIP) is an interesting technology that promises to complete with the ITU's H.32x family of specifications. Even though the SIP approach to conferencing is still immature, it is gaining acceptance in some portions of the vendor community, most notably the 3rd Generation Partnership Program (3GPP), a consortium aimed at the development of specifications for the next generation of mobile communications. As an IETF specification, SIP has the feel of other Internet protocols and offers some capabilities drawn from this domain not found in the ITU specifications. At the same time, SIP lacks some of the key capabilities found in H.323. In particular, quality of service issues are not handled by SIP at this time. Like H.323, SIP allows for interaction with public switched telephone networks.

### *3.3.1.15 Email*

A number of standard protocols and message formats exist for sharing of email messages. These include:

- Post Office Protocol (POP)
- Internet Message Access Protocol (IMAP)
- Simple Mail Transfer Protocol (SMTP)
- MIME (& S/MIME)

#### 3.3.1.15.1 Post Office Protocol

One of the two most popular email "client" protocols, POP exists to allow for downloading of email messages from a mail server to a user's computer. Unlike IMAP, which can operate with messages stored on a remote message store, POP's standard approach is "offline" operation in which messages are retrieved from the store and then deleted from this server.

#### 3.3.1.15.2 Internet Message Access Protocol

A more recent entry into the email protocol market, IMAP is suited to environments in which a user may wish to read his messages from different computers. Messages handled by an IMAP client can be handled in place on the message store that contains them so that messages do not end up being scattered across multiple machines (as might happen if only POP were used). The protocol also allows for management of server-based mailboxes, for server-based parsing of email messages, and for filtering of message

traffic between the server and the client (including selective retrieval of messages and message elements).

#### 3.3.1.15.3 Simple Mail Transfer Protocol (SMTP)

As the "basic protocol for Internet electronic mail transport", SMTP specifies the set of messages used both to transfer email from a client located on an end user's machine and to transfer messages between the "mail transfer agents" that make up the Internet email delivery infrastructure.

#### 3.3.1.15.4 RFC 2822, MIME, and S/MIME

RFC 2822 ("Internet Message Format") specifies the structure of text-only email messages. The Multipurpose Internet Mail Extensions (MIME) specifications extend this RFC to allow for the transmission of a broader range of data types (e.g., audio, images) via email. S/MIME provides a standard mechanism for providing for privacy, authenticity, and non-repudiation in MIME-based messages.

### *3.3.1.16 News Groups*

The standard protocols for newsgroups is the Network News Transfer Protocol (NNTP). This form of communication is analogous to a bulletin board (i.e., you pull information toward you rather than having pushed, as in email) and allows for threaded discussions (i.e., you see who said what when and what everyone said in response to that and then what people said about the responses and …).

### *3.3.1.17 Chat*

In low bandwidth situations, "chat" mechanisms that allow interactive text-based communications are of use. Protocols supporting this capability include:

- Internet Relay Chat (IRC)
- SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE)

#### 3.3.1.17.1 IRC

Internet Relay Chat is relatively old protocol used originally in the BBS arena to allow text-based, interactive communication between users of a bulletin board and is widely deployed today. In using IRC, an IRC client attaches to a specified IRC server that, in turn, can maintain connections with a number of IRC servers, thereby forming an IRC network. In addition to using "channels " to allow multiple (potentially thousands) of clients to share a common conversation, IRC provides mechanisms that allow direct client-to-client interactions that bypass the IRC servers

#### 3.3.1.17.2 SIMPLE

SIMPLE is a relatively new initiative of the IETF that proposes to standardize services for:

- "instant messaging" – the ability to conduct interactive, text-based communications between two or more parties, and
- "presence" – the ability to determine when a person with whom you might like to interact is on-line.

Based on IETF's Session Initiation Protocol (SIP) – a mechanism for allowing a user's networked devices to discover the location of other users' devices and to establish user-to-user communication sessions (text, voice, video) – this family of protocols appear to be poised to move onto center stage as a core element for establishing interactive person-to-person communications in future networked environments.

### *3.3.1.18 File Sharing*

Several protocols exist for transferring files between stations and for interactively sharing files. Among others, these include:

- File Transfer Protocol (FTP)
- IRC
- T.127

#### 3.3.1.18.1 File Transfer Protocol

File Transfer Protocol (FTP) is an old protocol for exchange of files between two computers. By abstracting the directory structures of both the client and server computers involved in the exchange, files can be downloaded from server to client or uploaded from client to server. In certain cases, a client located on a third machine can mediate this exchange. Recent updates to the protocol address its operation in IPv6 networks and describe how FTP can be extended with additional commands and how clients can explore which extended commands a given server supports.

#### 3.3.1.18.2 IRC

In addition to supporting group-oriented and person-to-person text-based communication, IRC provides client-to-client file transfer mechanisms.

#### 3.3.1.18.3 T.127

Part of the ITU's T. 120 family of protocols for data conferencing, T.127 builds on T.12x's multi-point communication services to allow for sharing of data files between two or more computers. Which participants in a conference receive transmitted files can be established dynamically.

### *3.3.1.19 Registry Server Interactions*

To date, ebXML's registry work looks like the most extensively defined. As noted above, the current favorites here are the UDDI and WSDL elements of generic web services. ebXML is moving toward richer mechanisms, of which registries are a central part, for facilitating business to business interactions. It should be noted that, to date, public registries based on these capabilities have been slow to gain a constituency, due in large part to a lack of control over the registries and associated security concerns. The move toward privately run registries may address these issues.

### *3.3.1.20 Dynamic Configuration*

A number of examples of how to package and download software from vendor sites exist today. Two of relevance to this issue are:

- Open Software Description (from Microsoft) and
- HAVi's code units.

While they each address different aspects of the software download problem, together they can be used to address the need for dynamic configuration (at least in the context of Java-based applications). Note that in both cases, a construct like the manufacturer's URL feature of Universal Plug-and-Plug can be used by a station's installation manager to find the web site for a device's vendor. Alternatively, vendors (or other sites hosting device-specific software) could be cataloged in the registry server network and located by query to the servers. Finally, it is interesting to note that using these techniques, it is possible to send to a server a description of the environment into which the software is to be installed and to have the server assemble the right package in real-time.

#### 3.3.1.20.1 Open Software Description

OSD was a Microsoft proposal to the W3C for standardized descriptions of the context of software packages and of how these contents relate one another. While it is not clear where the specification has gone in the W3C community, OSD is still used actively by Microsoft. In OSD, a "software package" is described with respect to the kind of environment (operating system, user's language, etc.) in which its associated software is intended to run. The components to which the OSD description refers are format-independent (e.g., could refer to Java code or to a Windows DLL or to application binary for a particular Unix variant running on a particular kind of processor). OSD is also quite flexible in describing where (on disk, on the net, etc.) a given software component in its description might be found.

#### 3.3.1.20.2 HAVi Code Units

Where the emphasis of OSD is on generic software package structure, the thrust of HAVi is on a way of delivering trusted Java code for use in intelligent audio-visual equipment. An approach for ensuring that downloaded packages are both authentic and uncorrupted is a centerpiece of this portion of the HAVi specification.

### 3.3.2 Recommendations

As we have noted several times already, the goal of this document is to foster discussion within the telemedicine community regarding how we want to build interoperability into our system. To this end, this chapter has proposed an abstract architecture for the construction of future systems and has provided an overview of current and emerging technologies that might be used to implement these systems. Given this background, we use the balance of this chapter to editorialize, making recommendations for specific interfaces that we believe need to agreed upon by the community and technologies for implementing these interfaces.

In making these recommendations, the following were used as guiding principles:

- Proprietary products should be avoided
- Present or likely market penetration is important
- Telemedicine systems should readily integrate with other systems used in healthcare environments

Note that several of the recommended technologies are still immature. While other, more stable, alternatives might be specified, the inclusion of these newer technologies should

be taken as a statement that these seem to be the coming technologies in their respective area.

| | |
|---|---|
| **Internal Communications** | |
| Internal Communications Bus | • Internet Protocol (without regard to underlying media) |
| Device Buses | • IEEE 1394 / Firewire (for wired)<br>• IrDA (for wireless) |
| Station Middleware | • Universal Plug-and-Play |
| **Internal Components** | |
| Station Registry | • TIA[1]-developed based on XML |
| **Patient Record Storage** | |
| EMR Structure | • GEHR |
| Record Storage Device Interfaces | • TIA-developed |
| **Medical Devices** | |
| Medical Device Interfaces | • IEEE 1073 for feature-rich devices<br>• POCCIC for "lightweight" devices |
| Medical Imaging Devices Interfaces | • DICOM |
| Standard Camera Interfaces | • PTP |
| **External Communications** | |
| Media | • POTS phone line<br>• Wired Ethernet (10/100BaseT)<br>• GSM (and successors) for wide-area wireless |
| Communications Manager | • TIA-developed |
| External Comms Manager | • TIA-developed |
| Session Manager | • TIA-developed |
| Registry Server Protocols | • TIA-developed based on web services |
| Session Management Protocols | • TBD |
| Medical/POC Device Remote Monitoring and Control Protocols | • TIA-developed (joint with IEEE 1073) |
| Imaging Device Protocols | • DICOM |
| Videoconferencing Protocols | • H.323 (for networked environments)<br>• H.324 (for PSTN environments) |
| Patient Record Exchange | • HL7 |
| Dynamic Configuration Protocols | • OSD |
| Person-to-Person Protocols | • SMTP (for mail transmission) |

[1] TIA – Telemedicine Interoperability Alliance

| | |
|---|---|
| | • POP3 *and* IMAP4 (for mail retrieval)<br>• RFC 2822, MIME and S/MIME (for mail encoding)<br>• SIMPLE (for interactive text messaging) |
| Other Protocols | • T.127 for file sharing |
| **User Interfaces** | |
| Media | • Visual: Independent but coordinated based on CCOW<br>• Voice: VoiceXML |
| User Interface Device Interfaces | • TIA-developed |
| **Security** | |
| User Authentication | • Kerberos |
| Access Control | • RAD |
| Encryption | • TBD |
| Data Authentication | • TBD |
| **System Intelligence** | |
| Contexts | • TIA-developed extensions to Liberty Alliance approach |
| Processing and Protocols | • Java |
| **Terminology** | |
| Semantics and syntax | • TIA-compiled synthesis/harmonization driven by content of other standards (e.g, DICOM, POCT, 1073) |

### *3.3.2.1 Internal Communications*

#### 3.3.2.1.1 Internal Communications Bus

IP has become the lingua franca of the digital communications world. Most emerging standards in this domain presume IP as the underlying protocol and while standards, like IEEE 1394, may gain a foothold in certain large markets, such as the home, even these standards typically have had work done to allow them to support an IP layer. Assuming that most of the station-external protocols will run over IP, having IP as the base internal protocol minimizes the amount of translation that needs to be done in moving from inside the station to outside and also makes its easier to implement virtual stations comprised of components that are widely distributed.

#### 3.3.2.1.2 Device Buses

The choice here does not seem clear cut; however, of the two logical choices for wired connectivity – IEEE 1394 and USB – USB looked like the right choice up until the last year or so. With delays in UWB 2.0, 1394 has gained ground and is appearing more commonly in computers. 1394 is also pervasive in the audio-visual industry. While both bus specifications were originally meant of connecting collocated components, new

extensions to the 1394 standard promise to make it usable as local area network and not just as a peripheral bus hanging off of a computer. The bandwidth of the most recent 1394 specification is significantly higher than that offered by the Whereas USB requires a root node to host the bus, 1394 allows for a peer arrangement in which any device capable of mastering the bus is allowed to do so. Given this, the vote went to 1394.

#### 3.3.2.1.3 Station Middleware

UPnP is recommended for communication between collocated components. The fact that it focuses on movement of data between peer components and does not constrain how these components are implemented allows significant flexibility for developers. In addition, as part of Microsoft's newest operating system, it is likely to enjoy a growing market presence (as opposed to some of its competing technologies, like Jini). At the same time, in as much as Microsoft has made the specifications for UPnP public, developers will not be locked into a proprietary solution set. While designed for ad hoc networking in which clients autonomously discover needed servers on the local network, the structure of UPnP allows for registry-style systems to be created. Finally, use of UPnP for certain functions does not preclude the use of other protocols (e.g., once UPnP is used to associate two components on a station, these components are free to interact using other protocols where doing so would be beneficial, as in bandwidth-intensive communications).

### *3.3.2.2 Internal Components*

#### 3.3.2.2.1 Station Registry

As the heart of the architecture, the station's registry will have much that is unique to this application domain; therefore, it is recommended that the telemedicine community develop the registry specification on its own using XML document technology as the underlying mechanism for encoding registry data.

### *3.3.2.3 Patient Record Storage*

#### 3.3.2.3.1 EMR Structure

The choice here is really between committing completely to HL7 and XML by choosing CDA or gaining greater flexibility through the user of GEHR. Because the focus of this particular recommendation is on a technology for developing the internal structure of an electronic medical record, buying into the power of GEHR in this realm seems the right choice, especially inasmuch as repository structures may vary widely with system employment concept.

#### 3.3.2.3.2 Record Storage Device Interfaces

It is not clear if any standards exists for allowing both the media and the repository systems into which medical records are to be placed to announce themselves to a system upon their installation and to then describe themselves. Given this, the current recommendation for this portion of the station is that the TIA develop these specifications.

### *3.3.2.4 Medical Devices*

#### 3.3.2.4.1 Medical Device Interfaces

In the non-imaging medical device world, the IEEE 1073 and POCCIC specifications already have a significant constituency and standard development organizations committed to their growth and maintenance. 1073 is recommended for the kinds of heavyweight devices that its already targets. POCCIC is recommended for use in "lightweight" devices where the overhead of 1073 would be prohibitive. Note that in both cases, the telemedicine community may end up developing, in cooperation with the applicable SDOs, the new device-specific application protocols for devices not yet addressed by these standards.

#### 3.3.2.4.2 Medical Imaging Device Interfaces

In the medical imaging device world, there is only one game in town – DICOM. Even with the variety of issues that the DICOM-community must face (e.g., achieving real plug-and-play assembly of imaging systems), proposing to go in a different direction would bring nothing but pain.

#### 3.3.2.4.3 Standard Camera Interface

With the explosive growth of digital photography over the last few years, a whole industry is springing up to service the needs of cyber shutterbugs. PTP appears to be the central protocol enabling many of these capabilities and, therefore, is recommended as the camera interface standard for the telemedicine community.

### *3.3.2.5 External Communications*

#### 3.3.2.5.1 Media

Three media are recommended as the basis for station-external communications. For the foreseeable future, standard wired phone lines will be the one universal constant in areas where phone service has long existed. In areas where it has not, GSM (and its successors) are poised to capture the wireless market. In high-bandwidth wired communications environments, irrespective of the mechanism used to access the Internet (e.g., router, DSL modem, or cable modem), the user side of these devices will continue to offer Ethernet connectivity (10/100 BaseT).

#### 3.3.2.5.2 Communications Manager

The set of services that the communication managers present to other components in the station are TIA-specific given that it is not clear that a standardized, message-oriented interface for this set of services exists yet in industry.

#### 3.3.2.5.3 External Communications Manager

Given the telemedicine station-specific nature of this component's operations, the recommendation is for this component is that its specifications be TIA-developed.

#### 3.3.2.5.4 Session Manager

As with the external communications manager, this component is intimately identified with the telemedicine station and should be specified by the TIA (with use being made of IETF QOS work when this is deemed mature enough).

#### 3.3.2.5.5 Registry Server Protocols

Because the application protocols associated with interacting with the registry server will be unique to the telemedicine community, it is recommended that this community develop them. Given the closeness of this work to ongoing activities in the XML and web services world, using web service capabilities (e.g., SOAP, UDDI) wherever possible is recommended in order to enable telemedicine to exploit, if possible, the infrastructure that is likely to develop to support web commerce.

#### 3.3.2.5.6 Session Management Protocols

For the reasons outlined above in the discussion on session management options, no specific recommendation is being made here.

#### 3.3.2.5.7 Med./POC Device Remote Monitoring & Control Protocols

There is relatively little that has been done in this area and the telemedicine community may be as experience as anyone in this arena. Since IEEE 1073 is beginning to address these issues, pursuing development of these specifications jointly seems like the most reasonable course of action.

#### 3.3.2.5.8 Imaging Device Protocols

Once again: DICOM rules, why fight it?

#### 3.3.2.5.9 Videoconferencing Protocols

This choice is much less clear. The Internet-based work of the IETF looks like it may, in time, carry the day simply because it is riding a wave of burgeoning Internet technologies and because the vision of computer-mediated communication services is quite compelling. Even so, the IETF work is still in its early stages – a point where the promise of what might be can differ significantly from what is actually developed. Given these things, it seems prudent to stick with the ITU specifications – H.3232 for networked environments and H.324 for PSTN – which, by the way, are by no means stagnant.

#### 3.3.2.5.10 Patient Record Exchange Protocols

As DICOM is for imaging, HL7 is for patient record exchange. Its constituency is so large that it is hard to imagine anything threatening its dominance in the foreseeable future.

#### 3.3.2.5.11 Dynamic Configuration Protocols

Microsoft's OSD is recommended as the mechanism for packaging downloadable software because of its generality (i.e., it's not language specific) and flexibility.

#### 3.3.2.5.12 Person-to-Person Protocols

For email messaging, the dominant standards are SMTP for mail transmission, POP3 *and* IMAP4 for mail retrieval and MIME and S/MIME for mail encoding. In the area of text

message, the choice is not completely clear, but given that a number of instant messaging activities are embracing it, SIMPLE seems like the right choice.

#### 3.3.2.5.13 Other Protocols

Since the ITU standards for videoconferencing are being recommended, T.127 (which is part of the H.32x family) is the obvious choice for sharing of data files.

### *3.3.2.6 User Interfaces*

#### 3.3.2.6.1 Media

For the reasons discussed above in the section of user interface components, it is not recommended that a specific user interface medium be established as the standard for the telemedicine community. At the same time, when visual (GUI-based) components are being incorporated into the station (especially as part of stand-alone applications that run in parallel with other applications on the station), CCOW capabilities also be implemented in order to allow for synchronization of context between independent applications. When voice-based interfaces are being built into a station, compliance with VoiceXML is recommended.

#### 3.3.2.6.2 User Interface Device Interfaces

Inasmuch as it is not clear whether a standard, device-independent mechanism exists for allowing a user interface to announce its presence and to describe its capabilities to a station, it is recommended that the TIA develop this specification.

### *3.3.2.7 Security*

#### 3.3.2.7.1 User Authentication

Kerberos is recommended as the mechanism for distributed user authentication. The assumption underlying this recommendation is that Internet-based authentication services (e.g., Microsoft's Passport service and those that will be compliant with Liberty Alliance specs) that appear to be emerging today (and that rely on this protocol) will succeed and become the standard mechanism for allowing unfamiliar parties to successfully identify each other.

#### 3.3.2.7.2 Access Control

One of the very good ideas to emerge from the CORBA community's healthcare work is the RAD specification. It is recommended because it allows for the creation of very sophisticated access control policies on any (software-based) resource in a system.

#### 3.3.2.7.3 Encryption and Data Authentication

Both the approaches that you choose for encrypting data links and the approaches that are chosen for data authentication are closely tied to the specific protocols chosen for other aspects of the station and system architecture. For instance, choosing HL7 means that you also choose to implement S/MIME for secure transmission of HL7 messages. Similarly, a commitment to IP-based communications will probably mean, in time, a commitment to IPSEC-based assurance. Given this, recommendations in this area are TBD.

### *3.3.2.8 System Intelligence*

#### 3.3.2.8.1 Contexts

Although it is always hard to tell with Microsoft, it looks like the Liberty Alliance approach to serving user-specific data may gain support from Redmond. Given its more generic approach and ability to support decentralized implementations, Liberty seems like a preferred approach and a reasonable base for building the telemedicine-centric contexts.

#### 3.3.2.8.2 Processing and Protocols

The primary motivation for recommending Java as the basis for developing processes and protocols is the desire to easily move code between platforms in a distributed station implementation and the body of capabilities that have been built up in the web and mobile worlds to accomplish just that.

### *3.3.2.9 Terminology*

#### 3.3.2.9.1 Semantics and Syntax

Making recommendations at this point would be a case of the tail wagging the dog. Just as with the encryption and authentication approaches discussed above, what you will choose is highly correlated with the other standards that you pick (e.g., if you adopt IEEE 1073, you have automatically bought into a large vocabulary related to vital signs monitoring, a specific way of encoding the elements of this vocabulary, and a way of communicating these concepts). Once "the rest" has been decided, this area will require work on the part of the telemedicine community to synthesize and harmonize the various concept sets and their representations.

# 4 Making Interoperability a Reality

This section proposes a path forward in the effort to make vendor-independent plug-and-play composition of distributed telemedicine systems a reality. It is one thing for one organization to propose an approach to achieve the kinds of goals described in this document; it is another thing altogether to achieve agreement within an industry regarding which approach is right for the industry. Reaching agreement within the telemedicine industry will take time. The ideas presented in this document must be studied and challenged. Partnerships must be established with other groups who share common interests. Demonstration systems must be built and "plugfests" held to flesh out and prove out the concepts described here. Mechanisms must be established for maintaining and evolving the "standards" that emerge from this work. Marketing efforts must be pursued to raise awareness and promote acceptance for the products of all of these efforts.

## *4.1 Background*

In March 1999, Sandia National Laboratories proposed to its Army program managers, the U.S. Army Telemedicine and Advanced Technologies Research Center (TATRC), that work be done to pursue the development of interoperability specifications for telemedicine stations. Planned as a three phase effort, the first product of this effort was delivered for clinical evaluation in the Fall of 2000. The focus of this "Build 1" device was on the infrastructure needed within a single station to support distributed system operation and on the mechanisms needed to support plug-and-play operation of medical devices. Build 2 in this plan was to focus on the issues associated with distributed, multi-station operation and Build 3 on replacing monolithic stations with distributed collections of components (such as might be found in a home networking environment). When funding for the final two phases of this effort were not forthcoming, TATRC provided enough funding to document the architecture concept (i.e., to produce this document that you are reading) and to begin socializing it in the community.

While the architecture presented in this document is based on the design patterns employed in the Build 1 design, there are differences between the two. In particular, this architecture attempts to address the requirements associated with systems in which stations are created completely from ad hoc collections of components (whereas the Build 1 design depended on a number of proprietary component interface specifications). In addition, Sandia's Build 1 system was built on CORBA and Java (although its design incorporated an "abstraction layer" that would allow for the replacement of CORBA with other middleware technologies), a feature that might or might not prove acceptable to the larger telemedicine community. For these reasons, while the majority of the concepts presented in this document are proven, implementation specifics still need to be hammered out through the development of prototypes that implement the proposed specification.

## *4.2 Compliance*

The architectural approach proposed in this document is meant to enable the dynamic creation of telemedicine stations and systems (of stations) from distributed, plug-and-play components and is meant to provide for a variety of services that might be used in a

telemedicine environment. As such, this document identifies a number of different interfaces that must be implemented if a system is to support all of the features specified. At the same time, nothing constrains a system developer to implement all aspects of this specification. As business needs dictate, vendors should be able to implement those interfaces that are useful to them and to forgo implementation of those parts that are not. To this end, the telemedicine community will need to establish "profiles" (see next section) that indicate which parts of the architecture must be implemented for a given utilization concept. Mechanisms that allow vendors to evaluate their conformance with given profiles should also be established.

In the broadest terms, the specification can be viewed as containing three sets of interfaces: station-to-device, station-to-station, and station-internal. The goal of the first interface – station-to-device communications – is to support vendor-independent, plug-and-play operation of devices (medical instrument, patient record, and user interface) that attach to a station. As the focus is on device interactions, at this level of interoperability, the implementer is not constrained in the way that his station is built other than to include whichever device bus approach(es) have been accepted by the telemedicine community and to support the application-level protocols defined for each of these devices. How the vendor's stations communicate with other stations or whether they communicate at all is entirely up to the vendor.

In the station-to-station communications interface, the focus is on creating stations that are capable of locating and contacting stations created by other vendors, on enabling stations to exchange information regarding the capabilities that they support, and on allowing stations to lease and control each other's resources. As with station-to-device communications, this level of interoperability levies no constraints on how developers build their stations. Whether the station is a monolithic design with a fixed set of resources or a fully distributed design with hot plugging resources is transparent to devices outside of the station itself. The intent of this design is to allow the telemedicine community to establish "open" networks with a minimum level of reengineering of existing systems. Because this set of interfaces allows for the control and monitoring of devices on remote stations, implementation of this set of interfaces implies the implementation of the application level protocols used interact with devices in the station-to-device communications.

The final set of interfaces address the station-internal communications (i.e., the way that the various components that make up at station communicate with one another). As such, these interfaces exist to allow for the ad hoc creation of stations using a variety of individual components, especially those scattered around a network. While it is likely that vendors implementing this set of interfaces will be creating systems that implement all three set of interfaces, there is nothing that requires that this be so. For example, it is easy to imagine a home-based care system that employs a self-contained suite of instruments (i.e., one in which the medical devices are all tightly integrated into single box), a patient record stored on a PC, and a user interface device (e.g., a computer-enabled TV) that are all interconnected by means of a home network. While this system would implement this final set of station-internal interfaces that would allow the different

resources on the network to discover each other and to federate to form a station, it is conceivable (though unlikely) that such a station would be operated as a stand-alone station (i.e., would not communicate with external stations), thereby obviating the need to implement all three interfaces.

## 4.3 Profiles

While vendors need the freedom to choose which parts of the architecture they wish to implement, allowing for complete freedom introduces a number of problems. In as much as the goal is to allow components and stations developed independently by different vendors to interact, a telemedicine product developer needs some sense of exactly what must be done in order to ensure that his products will function properly in a given operational setting. Likewise, consumers faced with a variety of product offerings will want to know which products from which vendors can be used together and for what purposes. It was for this very reason that, several years ago, the Technology Special Interest Group of the American Telemedicine Association identified telemedicine system "component labeling" as a key issue that needed to be addressed by the telemedicine community.

To address these needs, we suggest that a series of profiles be established that represent the various operational concepts that the telemedicine community would like to see supported and that specify which parts of the architecture and its associated specifications must be implemented in order for a vendor to assert that his product is "compliant". As a starting point for discussions, we will use six areas and the associated list of attributes shown in Figure 80 to characterize a station within a telemedicine system:
As used in the figure, the attributes are understood as follows:

- A station may support person-to-person communications. If it does, the methods employed can include interactive communications (e.g., teleconferencing/videoconferencing), store-and-forward communications (email, voice messaging), or both.
- Stations that host medical devices may employ devices that are integrally bound to the station, devices that attach in plug-and-play fashion, or both. If the station allows other stations to access these devices remotely, it is assumed that this remote access includes remote reading of data produced by the devices. It may or may not include remote control of these devices by the other stations. Accordingly, a station accessing a remote station that hosts one or more medical devices may read the data generated by the devices, may remotely control the devices, or both.
- If a station incorporates patient record data storage, this storage can be dedicated to one user at a given time or may be available to multiple users simultaneously. In addition, data in a station's repository may be stored on fixed media (e.g., hard drives embedded in the station) or on removable media. If the station allows other stations to read from and write to its patient record repositories, this may be done in real-time (e.g., for data streaming from a medical device) or in off-line/store-and-forward mode.

| | | Stand-alone Station | Tele-monitoring Station | Videoconferencing Station | Traveling Nurse Station | Minimal Patient Station | Minimal Caregiver Station | Classic Patient Station | Classic Caregiver Station | Bedside Hub | Full Patient Station | Full Caregiver Station | Patient Record Server | Basic Patient Card Reader | Net-ready Card Reader | Patient Record Browser | Net-ready Device | Protocol Server |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Person-to-person Communications | Interactive | | | X | | X | X | X | X | | X | X | | | | | | |
| | Store-and-forward | | | | | | | | | | X | X | | | | | | |
| Medical Devices | Integral | | X | | | | | X | | | X | | | | | | X | |
| | Plug-and-Play | X | | | X | X | | | | X | X | | | | | | | |
| | External Communications | | | | | X | X | X | X | X | X | X | | | | | X | |
| | Remote Control | | | | | | | | | X | X | X | | | | | X | |
| Patient Record | Single User | X | X | | | | | X | | | | | | X | X | | | |
| | Multi-user | | | | X | | | | X | | X | X | X | | | | | |
| | Fixed Media | X | X | | X | | | X | X | | X | X | X | | | | | |
| | Removable Media | | | | | | | | | | X | | | X | X | | | |
| | Store-and-forward | | X | | X | | | X | X | | X | X | X | | X | X | | |
| | Real-time | | | | | | | | X | | X | X | X | | X | | | |
| User Interface | Integral | X | X | X | X | X | X | X | X | | X | X | | X | | X | | |
| | Plug-and-Play | | | | | | | | | X | X | | | | | | | |
| Dynamic Configuration | Platform | | | | | | | | | X | X | X | X | | X | X | | X |
| | User Environment | | | | | | | | | X | X | X | | | | | | X |
| Addressing | Registration | | | | | | | | | | X | X | X | | | | | X |

**Figure 80. Candidate Profiles**

- A station may or may not incorporate a user interface. If it requires a user interface, it may have one that is integral to the station, one that is added dynamically (e.g., via a PDA), or both.
- In terms of station configuration, a station may be static with respect to the functions that it delivers or may be dynamically configurable. Likewise, the operating environment that it presents to its users may be fixed or may change dynamically as different users access its capabilities. For these reasons, this architecture allows for the existence of servers that provide on demand access to dynamically loadable processes, protocols, and contexts.
- Finally, a station may work through direct addressing (e.g., external stations interact with it by specifying its URI or a corresponding alias) or through advertising its existence and discovering the existence of other stations through the use of registry servers.

### 4.3.1 Stand-alone Station

**Description:** Stations in this profile are characterized by a "platform" that hosts one or more plug-and-play medical devices, an integral user interface, and (potentially limited) patient record storage intended for a single user.

**Utilization Concepts:**
1. A patient monitors his own progress at prescribed intervals using the various devices supported by the station. Readings collected in this way are archived in the station. Periodically, a caregiver contacts the person by phone and the data contained in the station is retrieved reading-by-reading and conveyed verbally to the caregiver who transcribes this data into the person's medical record.
2. Same as above with the exception that a traveling caregiver periodically visits the person and retrieves the data directly from the station and transcribes the data into the patient's record.

**Examples:**
1. Palm Pilot / PocketPC with wireless spirometer
2. Bedside "alarm clock" with IR connection to glucose monitor and blood pressure monitor that, in addition to functioning as a regular clock, displays and stores readings gathered from these devices

**Basis of Compliance:**
1. Use of TIA-supported device bus
2. Use of TIA-compliant interfaces for whatever medical devices (including imaging devices and general purpose cameras) the station is designed to support

### 4.3.2 Tele-monitoring Station

**Description:** Stations in this profile are characterized by a "platform" that hosts one or more integral medical devices, an integral user interface, and (potentially limited) patient record storage intended for a single user. They also provide for retrieval by

a remote station of readings logged into the station's patient record.

**Utilization Concepts:**
1. A patient monitors his own progress at prescribed intervals using the various devices supported by the station. Readings collected in this way are archived in the station. Periodically, a caregiver contacts the patient station and, after proper user authentication, uploads from the patient station all readings generated since the last upload.

**Examples:**
1. Dedicated platform with integral display to which devices attach and which attaches to phone line and to phone (in answering machine fashion)
2. System based on video game unit (e.g., Sony Playstation or Microsoft Xbox) that attaches to a TV's display and to medical devices and that accesses the Internet via a cable modem connection

**Basis of Compliance:**
1. Support for TIA external communications manager
2. Support for TIA registry communications
3. Support for TIA session management protocols
4. Use of TIA-compliant user authentication mechanisms
5. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)

### 4.3.3 Videoconferencing Station

**Description:** Stations in this profile are characterized by a platform that incorporates a dedicated user interface (display, camera, microphone, etc.) and provides for interactive video communications with a remote station.

**Utilization Concepts:**
1. A specialist on contract to a rural clinic interacts with the clinic's staff via videoconference link. Using a PDA-like device equipped with a wireless transceiver, the specialist is always available at a moment's notice to support a "curbside consult".
2. A fitness coach on staff at the HMO uses videoconferencing to make his weekly visits with his various lifestyle management groups.

**Examples:**
1. Typical videoconferencing box that connects to dedicated external monitor, cameras, microphone, speakers, and control unit and that can support a range of communications media (phone, dedicated line, etc.)
2. Touch-interface, wireless, tablet computer with integrated interface components

**Basis of Compliance:**
1. Support for TIA external communications manager
2. Support for TIA registry communications
3. Support for TIA session management protocols

4. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)

### 4.3.4 Traveling Nurse Station

**Description:** Stations in this profile are characterized by a compact platform capable of hosting a range of portable, plug-and-play medical devices, an integral user interface, and patient record storage intended for multiple patients. Clinical operations performed with the device are done in stand-alone mode (i.e., the station is not connected to other stations while be used for clinical functions). As needed, the station can be connected to remote stations for the purpose of synchronizing medical records on the nurse station and the remote station.

**Utilization Concepts:**

1. A nurse going to service a community downloads from the office server to the nurse station both the records for those patients scheduled to be seen and the master patient index for the patient record system. During the course of the day, records are opened and closed as needed and readings from devices and notes (text or voice) added by the nurse are entered into the currently selected record. If required, the nurse creates skeletal records for new patients and adds readings and notes to these records. At the end of the day, the nurse station is attached to the office network and the office record server is updated with the information generated by the nurse station.
2. A nurse carries a minimal nurse station into homes already equipped with medical devices and record storage. Once powered on, the station identifies the other devices in the house, synchronizes records as needed, and then leases the devices in the home for use during the visit. Recordings generated by these devices during the session are logged both to the local storage and to the traveling nurse station's patient record.

**Examples:**

1. Laptop PC with external "hub" that hosts various medical devices, keyboard-/mouse-driven graphical user interface, hard drive-based record storage, and PC Card-based wireless LAN connectivity.
2. Wearable PC with microphone and earpieces, wireless connection to medical devices, medical devices with integral "start/stop" controls, voice-driven command interface, solid-state data storage, and wireless connection to local area networks.

**Basis of Compliance:**

1. Use of TIA-supported device bus
2. Use of TIA-compliant interfaces for whatever medical devices (including imaging devices and general purpose cameras) the station is designed to support

3. Use of TIA-compliant user authentication mechanisms
4. Support for TIA external communications manager
5. Support for TIA registry communications
6. Support for TIA session management protocols
7. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)

### 4.3.5 Minimal Patient Station

**Description:** Stations in this profile are characterized by a platform that hosts one or more integral medical devices that can be monitored in real-time by a remote station, an integral user interface, and interactive, person-to-person communications. Intended for real-time use, this station does not provide for patient record storage.

**Utilization Concepts:**

1. A caregiver in a rural clinic examines patients with a suite of instruments that attach (by wire and by wireless link) to a base station mounted on the exam room wall. Integral to this station are a digital display, a camera, microphone, and speakers. The display is used both for display of digital data generated by the various instruments and for video-conferencing with remotely located specialists. Data collected by the station can be selectively forwarded to a remote specialist for real-time viewing.
2. A patient under the supervision of a home care agency receives periodic televisits from one of the agency's nurses. In additional to monitoring vital signs and other selected physiologic parameters, sessions usually involve some degree of discussion between the nurse and the patient regarding overall emotional state and other aspects of well-being.
3. A paramedic unit equipped with a portable version of this sort of station often carries the station with them to the accident scene. Live connection, including video feed from a sub-palm-sized video camera, to a supervising ER allows the ER docs to coach the paramedics through difficult cases.

**Examples:**

1. Videophone with external interface for medical devices.
2. Custom box containing tightly integrated flat panel display, camera, microphone, and speakers and integral, cable-connected blood pressure monitor, thermometer, weight scale, ECG, and built-in modem.

**Basis of Compliance:**

1. Use of TIA-supported device bus
2. Use of TIA-compliant interfaces for whatever medical devices (including imaging devices and general purpose

cameras) the station is designed to support
3. Use of TIA-compliant user authentication mechanisms
4. Support for TIA external communications manager
5. Support for TIA registry communications
6. Support for TIA session management protocols
7. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
8. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)

### 4.3.6 Minimal Caregiver Station

**Description:** Stations in this profile are characterized by a platform that permits remote monitoring of data coming from one or more medical devices on a remote station and includes an integral user interface and interactive, person-to-person communications.

**Utilization Concepts:**
1. An on-call specialist is able to receive a "call" from a nurse in a critical care unit and, using the caregiver station, to remotely monitor the information being generated from a range of instruments at the bedside where the nurse is working and to interact with the nurse by audio and/or video link throughout this process.
2. Same as 1. but for remotely located paramedic unit.

**Examples:**
1. PocketPC with "high-bandwidth" wireless Internet connection and videoconferencing add-in card
2. HAVi-enabled television with videoconferencing-compliant cable modem

**Basis of Compliance:**
1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols
5. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)

### 4.3.7 Classic Patient Station

**Description:** Stations in this profile are meant to address the features of patient stations are they are typically built today. Stations fitting this profile are characterized by a platform that hosts one or more integral medical devices that can be remotely monitored in real-time by remote stations, an integral user interface, interactive person-to-person communications, and patient record storage intended for a single user that can be

accessed for reading by remote stations.

**Utilization Concepts:**

1. A patient monitors his own progress at prescribed intervals using the various devices supported by the station. Readings collected in this way are archived in the station. Periodically, a caregiver contacts the patient station and, after proper user authentication, uploads from the patient station all readings generated since the last upload. An interactive session consisting of both patient-caregiver interaction and interactive collection of various readings follows.

**Examples:**

1. Custom box containing tightly integrated flat panel display, camera, microphone, and speakers and integral, cable-connected blood pressure monitor, thermometer, weight scale, ECG, built-in modem, and hard drive-based data storage.

**Basis of Compliance:**

1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols
5. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)
7. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)

### 4.3.8 Classic Caregiver Station

**Description:** Stations in this profile are meant to address the features of caregiver stations as they are typically built today. Stations fitting this profile are characterized by support for interactive person-to-person communications, the ability to monitor in real-time data generated by instruments on a remotely located station, an integral user interface, and patient record storage for multiple patients, and the ability to interact with (browse and upload) patient records stored on remote stations.

**Utilization Concepts:**

1. A home care nurse spends the day “visiting” a number of patients in her charge. As she places a call and it is answered, her station checks with the remote station to determine if any new readings have been generated since the last encounter and, if so, uploads these to the nurse’s patient record database. After reviewing the data and talking with the patient, she directs the patient to use one or more of the devices shown to be available at the patient’s location. Data generated by these actions are

automatically displayed on the nurse's station and simultaneously loaded into the patient's records on the station. Where appropriate, data can be analyzed in real-time (e.g., to detect trends in blood pressure) using processes local to the nurse's station.

**Examples**

1. A desktop PC with an add-in video conferencing card and custom software

**Basis of Compliance:**

1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols
5. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)
7. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)

### 4.3.9 Bedside Hub

**Description:** Stations in this profile are characterized by a barebones platform that is augmented in plug-and-play fashion with user interface components and medical devices and that is capable of dynamic configuration of both its operating environment (e.g., software used to control attached devices) and its user environment (e.g., the protocols implemented in support of a given user and the external patient record repositories to which the station connects while a given user is present).

**Utilization Concepts:**

1. A CCU bed contains an integral platform that presents a suite of standardized device interfaces to the external world as well as a standardized way of connecting to a local area network. The bed constantly monitors its environment for the presence of devices to which it can connect. Under user supervision, the bed establishes associations with specified medical devices and user interface devices, in the process retrieving software components that it needs in order to support the devices with which is has established associations. Subject to context information retrieved about users that are present, the bed is able to configure itself with the users' protocols of choice and to establish connections to external repositories and external processing resources.

**Examples:**

1. Custom box with wireless and wired interface ports for medical device and user interface communications and PCMCIA slot that can host interfaces cards for various

types of wired and wireless network.
2. OEM board with fixed device and network interface connections that can be integrated into other products, such as hospital beds or wall-mount units.

**Basis of Compliance:**
1. Use of TIA-supported device bus
2. Use of TIA-compliant interfaces for whatever medical devices (including imaging devices and general purpose cameras) the station is designed to support
3. Use of TIA-compliant user interface device interfaces
4. Use of TIA-compliant user authentication mechanisms
5. Support for TIA external communications manager
6. Support for TIA registry communications
7. Support for TIA session management protocols
8. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
9. Use of TIA-compliant medical device remote control protocols (and associated data authentication and encryption protocols)
10. Use of TIA-compliant dynamic configuration protocols (and associated data authentication and encryption protocols)

### 4.3.10 Full Patient Station

**Description:** Stations in this profile incorporate most of the capabilities defined in this architecture. These include:

- both interactive (video- and audio-conferencing) and store-and-forward (email, voicemail, videomail) person-to-person communications
- the possibility for both integral and plug-and-play medical devices which can be monitored as well as controlled by remote stations
- local, multi-person patient record storage that can be browsed or bulk uploaded from remote stations and the ability to read from and write to patient record cards
- the use of integral user interface capabilities, of dynamically added interfaces, or both
- the ability to dynamically load new software components into the station to handle changing device suites
- the ability to dynamically load new protocols and establish external connections to support changing user and patient populations
- registration of the station with registry servers to allow other stations to discover the station's advertised capabilities and network address

**Utilization Concepts:**
1. A cart carrying both the patient station and a wide range

of medical instruments is rolled where needed in the nursing home. Once in the room, the station establishes contact with the room's network access point and based on context information in the access point establishes external connections to the proper patient record in the nursing home's patient record server and displays this record automatically on the station's screen. As the nurses attach devices to the station, the software components needed to control the instruments are dynamically loaded in to the station. Reading generated by the station are displayed on its screen and automatically forwarded to the patient record server. As needed, the nurses can place a video call to the resident physician who is able monitor their operations in real-time and, if required, can remotely configure and operate the devices.

2. In the home, a well-equipped patient station acts as a medical "portal" allowing a person to interact with a variety of caregivers. As needed the station's capabilities can be extended with devices purchased by the person or rented as needed from medical supply houses.

**Examples:**

1. PC-based system with built-in device bus interfaces, network interface, integral videoconferencing components (camera, speakers, etc.), and card reader.
2. Home network-based system using TV for interface, local computing resources for processing and storage, home gateway for Internet communications, network access points medical device communications, and net-ready "medical peripherals".

**Basis of Compliance:**

1. Use of TIA-supported device bus
2. Use of TIA-supported internal communications bus
3. Use of TIA-compliant interfaces for whatever medical devices (including imaging devices and general purpose cameras) the station is designed to support
4. Use of TIA-compliant user interface device interfaces
5. Use of TIA-compliant record storage device interfaces
6. Use of TIA-compliant user authentication mechanisms
7. Support for TIA external communications manager
8. Support for TIA registry communications
9. Support for TIA session management protocols
10. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
11. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)
12. Use of TIA-compliant person-to-person communication

protocols (and associated data authentication and encryption protocols)
13. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)
14. Use of TIA-compliant dynamic configuration protocols (and associated data authentication and encryption protocols)
15. Use of TIA-compliant registry server protocols

### 4.3.11 Full Caregiver Station

**Description:** Stations in this profile incorporate most of the capabilities defined in this architecture and differ from the full patient station only in the lack of medical devices at the caregiver station.

**Utilization Concepts:**
1. A patient looking for a new doctor searches the telemedicine registry servers to find one that meets certain criteria. After "dialing" the doctor, the patient is connected with a nurse practitioner whose station scans the patient's station to discover what capabilities are local to the station and to learn where to locate the patient's records. Creating a new record for the patient at her station, the nurse links the existing records into hers. Local processing routines analyze the record, flagging items of interest and creating a summary for her review. After a short discussion, she then begins the balance of the encounter, moving into current complaints and, as needed, controlling devices on the patient's station to collect current readings.

**Examples:**
1. Network computer with distributed computing resources.

**Basis of Compliance:**
1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols
5. Use of TIA-compliant medical device remote monitoring protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant videoconferencing protocols (and associated data authentication and encryption protocols)
7. Use of TIA-compliant person-to-person communication protocols (and associated data authentication and encryption protocols)
8. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)
9. Use of TIA-compliant dynamic configuration protocols

(and associated data authentication and encryption protocols)

10. Use of TIA-compliant registry server protocols

### 4.3.12 Record Server

**Description:** Stations in this profile are characterized by a platform that permits remote storage and retrieval of records for multiple patients. As needed, the platform can download new components needed for handling data types that are new to the server and for handling different data exchange protocols used to move data between the server and a range of external devices.

**Utilization Concepts:**

1. A clinic stores all of their records on a single, high reliability server maintained at the clinic. Doctors doing rounds at area hospitals are able to automatically forward data from all encounters to their records in the clinic's server. In like fashion, clinic staff away from the clinic are able to remotely review records stored on the server.
2. A patient subscribing to a medical record service provides the service with the names of all doctors and institutions with which they have been associated. In turn, the service provides the patient with a record card that contains the patient's context, including the patient's account number with the service. The service then contacts each of the noted providers and gains access rights to the patient's records on each of these servers. When the patient visits a caregiver, the context information read from the patient record card directs the caregiver or patient station to a single server that acts as the "front-end" for all of the other servers that store the patient's information. To the querying station, it appears at first blush that all records for the patient are stored on this single station. As needed, the user at the querying station can search further to determine the actual source of each record in the virtual record presented by the record server.
3. All records for a family are stored on a record server device located on the family's network. As with commercial systems, this repository serves records for multiple patients. Whenever a family member visits a caregiver from home, the local server maintains a record of the transaction. When the family members visit a caregiver in person, the context on the patient card that they present directs the caregiver's station to send a copy of the encounter record to the family's personal record server.

**Examples:**

1. Server-class PC running dedicated database software

2. Distributed database running across distributed machines front-ended by multiple, load-balancing gateway machines, presenting a record server interface to the outside world.
3. Net-ready database "appliance" that can be added to a home network in "plug-and-play" fashion.

**Basis of Compliance:**
1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols
5. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant dynamic configuration protocols (and associated data authentication and encryption protocols)
7. Use of TIA-compliant registry server protocols

### 4.3.13 Basic Patient Card Reader

**Description:** Stations in this profile are characterized by a platform that permits browsing and editing of records contained on a patient record card. These devices are self-contained, having their own native user interfaces and no connections to the outside world.

**Utilization Concepts:**
1. A hand-held device with wireless card reader with text and audio user interface elements is used by battlefield medics to read the "electronic dog tags" of injured personnel and to annotate the record before the personnel are evacuated.
2. Paramedics use a card reader to review critical information contained on a "medic alert" bracelet worn by certain patients.

**Examples:**
1. PDA with built-in reader

**Basis of Compliance:**
1. Use of TIA-supported device bus
2. Use of TIA-compliant record storage device interfaces

### 4.3.14 Net-ready Card Reader

**Description:** Stations in this profile are characterized by a platform that permits remote stations to browse and edit records contained on a patient record card. As needed, these devices can download software components needed for reading record types that are not native to the reader.

**Utilization Concepts:**
1. A wall-mounted reader in an examine room provides other station components in the room with access to the records contained on the patient's record card. When a protocol leasing the reader identifies the need for a new

record type (i.e., one not known by the reader), the reader can be directed to download the software components needed to support operations with this new record type.

**Examples:**
1. Custom box with interface to local area network.

**Basis of Compliance:**
1. Use of TIA-supported internal communications bus
2. Use of TIA-compliant record storage device interfaces
3. Use of TIA-compliant user authentication mechanisms
4. Support for TIA external communications manager
5. Support for TIA registry communications
6. Support for TIA session management protocols
7. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)
8. Use of TIA-compliant dynamic configuration protocols (and associated data authentication and encryption protocols)

### 4.3.15 Patient Record Browser

**Description:** Stations in this profile are characterized by a platform that permits browsing and editing of remotely stored patient records. As with other patient record devices, when needed, these devices can download software components needed for reading and writing new record types. Depending on the platform, the browser can also output various readings in different formats (e.g., text can be converted to audio).

**Utilization Concepts:**
1. At the end of each day, a doctor reviews the records for each of the day's encounters, annotating them as needed. While she usually does this at the desk in her office, she can also do this from her computer at home and, to a more limited extent, from her palmtop computer using a wireless Internet connection.
2. A patient subscribing to a patient record service uses software provided by the service that allows his virtual record to be viewed in a variety of different ways (chronologically, by type of problem, by providing organization, etc.) and that permits him to print reports corresponding to any of these views.

**Examples:**
1. PC with custom patient record software.
2. PC with standard web browser
3. Cell phone with audio "browser" interface that supports quick access to "hot" data or fuller browsing of the full record.

**Basis of Compliance:**
1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols

5. Use of TIA-compliant patient record exchange protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant dynamic configuration protocols (and associated data authentication and encryption protocols)

### 4.3.16 Net-ready Device

**Description:** Devices in this profile consist of medical instruments designed to connect directly to a local area network. These devices may be monitored locally and/or remotely and controlled in the same way.

**Utilization Concepts:**
1. A weight scale with integral controls and display reports its results over the local network to a patient station that is also attached to the net.
2. A band-aid style sensor applied to a patient's forehead reports temperature and various chemistries to an associated network transceiver that powers the sensor by periodically bathing it in RF radiation.

**Examples:**
1. A thermometer that communicates via IR link to a network access point that bridges the IR link to a network.
2. A digital video camera with a built-in IP networking capability that allows it to be addressed directly by other network devices.

**Basis of Compliance:**
1. Use of TIA-supported internal communications bus
2. Use of TIA-compliant device interfaces for whatever class the device belongs to

### 4.3.17 Protocol Server

**Description:** Devices in this profile consist of a platform capable of downloading protocols and processes to requesting stations, of locally executing protocols and processes on behalf of a remote station, or both.

**Utilization Concepts:**
1. A remote station requests a protocol to support a given kind of medical device that has been added to the station. Based on these criteria, the server identifies three possible candidates and returns to the station descriptions of each. When the station selects one, that protocol is downloaded to the station.
2. A remote patient station that hosts an ultrasound probe leases image-processing services from a company that whose research has yielded the most powerful analytical tools of their kind. As data from the probe is streamed to the company's server, it is run through various processes according to parameters established by the lease and the

results returned in real-time to the station.

**Examples:**
1. Server-class PC with Internet connectivity.
2. Server-class PC operating on a local area network providing specialized services to local stations operating on the same network.

**Basis of Compliance:**
1. Use of TIA-compliant user authentication mechanisms
2. Support for TIA external communications manager
3. Support for TIA registry communications
4. Support for TIA session management protocols
5. Use of TIA-compliant dynamic configuration protocols (and associated data authentication and encryption protocols)
6. Use of TIA-compliant registry server protocols

## 4.4 Moving Forward

Given this document as a starting point, we propose the following steps be taken towards the establishment of community-accepted specifications:

- Circulate the document within the community for initial comment and then incorporate suggestions where possible and compile a list of issues where points of disagreement need to be worked out.
- Convene a "summit" of groups interested in working on the advancement of specifications that support the concepts described in this document. The goal of the summit will be to discuss the issues raised during the comment period and to lay plans for doing prototype implementations of the specification. One of the key products of this meeting should be the establishment of a task force that will shepherd this work to its completion.
- Conduct a series of development/demonstration cycles in which various parts of the architecture are specified, proven out, and revised as needed to address issues found in the specifications.
- Release to the general telemedicine community the first proven set of specifications. This will include both the revised version of the document that you are reading as well as a number of companion specifications that address specific interfaces within the station (e.g., how a blood pressure monitor interacts with the rest of a station).

In the course of pursuing the various development/demonstration cycles, a key concern will be the rights to intellectual property generated by these efforts. Given this, the summit should focus on reaching agreement regarding the approach to be taken in addressing this issue. One reasonable approach would be to agree that everyone contributing to the development effort will make available to the rest of the developers that portion of the code that they develop that relates to the implementation of the interfaces within the specification. This code might be freely shared among all such participants or licensed to members within the development group on a "value of contribution" basis.

As noted earlier in this document, one of the goals of this effort is to avoid reinventing the wheel. Much has already been done within the medical standards arena that is of value to the telemedicine community. To this end, as the groups that are to carry this work forward are assembled, it is vitally important that the groups ow ning these related standards be engaged. Among others, these include ACR/NEMA for the DICOM standard, IEEE 1073 for medical device interactions (including the POCCIC work of which they are stewards), and HL7 for its work in medical data exchange standards.

## 4.5 A Proposed Set of Specifications

While this document provides the overview for the architecture and proposes its essential structure, a collection of more narrowly focused documents that address specific portions of this specification is needed in order to make this a usable interoperability architecture. For example, while this document proposes the general pattern that is to be used in connecting a medical device to the rest of the station, it says nothing about how a specific type of device, such as a blood pressure monitor, will be handled. For this, a specification that identifies the device-unique messaging structure of blood pressure monitors needs to be created (note "creating" that this might be nothing more than specifying how another existing specification will be employed in this architecture).

Given the scope of the architecture as addressed in this document, the following set of constituent specifications is proposed.

1. ***General Documents***
   These documents provide an overview of the system architecture and contain information that spans each of the other functional areas within a station. In addition to the Concept Description and Architecture Overview Document, this set of documents will contain detailed specifications for station internals (e.g., structure and protocols for the registry, internal communications bus, contexts,) and will address things like vocabulary and common services (e.g., security).
2. ***Device Bus Documents***
   These documents address the low-level communication mechanisms used within a station (or its constituent components) to support plug-and-play operation of certain devices, such as medical instruments, patient record storage media, and certain user interface devices
3. ***Medical Device-Oriented Documents***
   These documents describe the application-level interfaces for the various medical devices (e.g., glucose meters, cameras and scopes) that are supported by the telemedicine architecture
4. ***Communications-Oriented Documents***
   These documents focus on the application-level interfaces for the various types of channels (corresponding to each of the station-to-station protocols supported) and communication managers (corresponding to each of the communication devices supported) and for the session manager (corresponding the different securit y and quality of service mechanisms supported).
5. ***Patient Record Documents***

These documents capture the application-level interfaces for the "native" patient record structure supported by the architecture, for the distributed patient record systems, and for the mapping between external data exchange protocols (e.g., HL7) and the native application interface. The documents also address the interface between plug-and-play patient record media and a station.

6. ***Processing-Oriented Documents***
   These documents specify the interfaces for various process components that can support architecture-compliant telemedicine stations. Among other things, these can include statistical processing (e.g., trending in data), compression functions, and person-to-person communications (e.g., email, voicemail) functions.
7. ***Protocol-Oriented Documents***
   These documents specify the interfaces for various process components that can support architecture-compliant telemedicine stations. Among other things, these can include statistical processing (e.g., trending in data), compression functions, and person-to-person communications (e.g., email, voicemail) functions
8. ***User Interface-Oriented Documents***
   These documents deal with the application interfaces that various user interface components present to the other components within a station. It also addresses various sorts of user interface frameworks that support on-the-fly customization of the user interface.
9. ***Infrastructure Documents***
   These documents address the application interfaces for those service elements that a required to make a collection of telemedicine stations function as a seamless network (registry servers, public key infrastructure elements, trusted time sources, and vendor software services).

## *4.6 Growing the Specifications*

In growing and proving out the specifications, a number of capabilities need to be demonstrated. As these cannot all be done at once, we suggest the following plan:

**Phase 1**

Work on those aspects of the architecture that relate to the local operation of medical devices

A. Prove out the application interfaces for a representative set of medical devices, excluding imaging devices
B. Demonstrate the application interfaces for a representative set of imaging devices
C. Demonstrate plug-and-play operation of all of these devices

**Phase 2**

Work on basic station-to-station operations

A. Demonstrate the ability to explore the capabilities of a remote station
B. Demonstrate the leasing, operating, and vacating of medical device capabilities of a remote station
C. Add videoconferencing
D. Add other person-to-person communications

**Phase 3**

Work on the user interface and processing aspects of the architecture

A. Prove out the application interfaces for user interface components created to date
B. Demonstrate plug-and-play operation of selected user interface devices and web presentation of user interfaces
C. Prove out the application interfaces for processing components created to date

**Phase 4**

Move towards component-based station implementations

A. Demonstrate operation of internal communications bus
B. Prove out the use of protocols and subscription interfaces to handle the interconnection of resources within a station
C. Demonstrate a distributed station implementation in which station components are capable of announcing their presence and being added dynamically to the station under the control of protocols

**Phase 5**

Add patient record capabilities to the stations

A. Prove out the native application-level interfaces for the patient record repository
B. Demonstrate plug-and-play operation of patient record devices
C. Demonstrate the ability to employ patient record services provided by one or more remote stations
D. Demonstrate the ability to negotiate and support a range of patient record exchange protocols (e.g., COAS, HL7, DICOM)

**Phase 6**

Incorporate security mechanisms into the stations

A. Demonstrate the ability to identify and authenticate a user attempting to access a station either locally or remotely and to selectively grant or deny access to services and data based on this information
B. Demonstrate the ability to negotiate and secure station-to-station communications and to negotiate and support station-to-station quality of service
C. Demonstrate the ability to negotiate and secure station-internal communications and to negotiate and support station-internal quality of service

**Phase 7**

Work on advanced station configuration capabilities

A. Demonstrate the ability to temporarily and permanently retrieve and install software components needed to support the operation of other components on a station
B. Demonstrate the ability to dynamically configure a station's operation based on patient and caregiver context data

**Phase 8**

Add station administration functions to system

A. Demonstrate the ability of a station to add itself to or remove itself from a registry
B. Demonstrate the use a network of registry servers to provide the ability to locate stations by name and by station attributes

**Phase 9**

Demonstrate the ability to operate in a multi-station (i.e., 3 or more simultaneously) mode and to dynamically add and remove stations from multi-station conferences

### 4.6.1 Phase 1: Medical Device Operations

This phase of the work addresses those aspects of the architecture that relate to the local operation of plug-and-play medical devices (i.e., remote control of devices is not part of this phase). Figure 81 identifies which elements of the architecture are the focus of this phase concentrates. The plan for this phase is:

- Prove out the application interfaces for a representative set of medical devices, excluding imaging devices
- Demonstrate the application interfaces for a representative set of imaging devices
- Demonstrate plug-and-play operation of all of these devices

When complete it should be possible to plug devices from a number of medical device vendors into telemedicine stations from a number of different telemedicine system vendors.

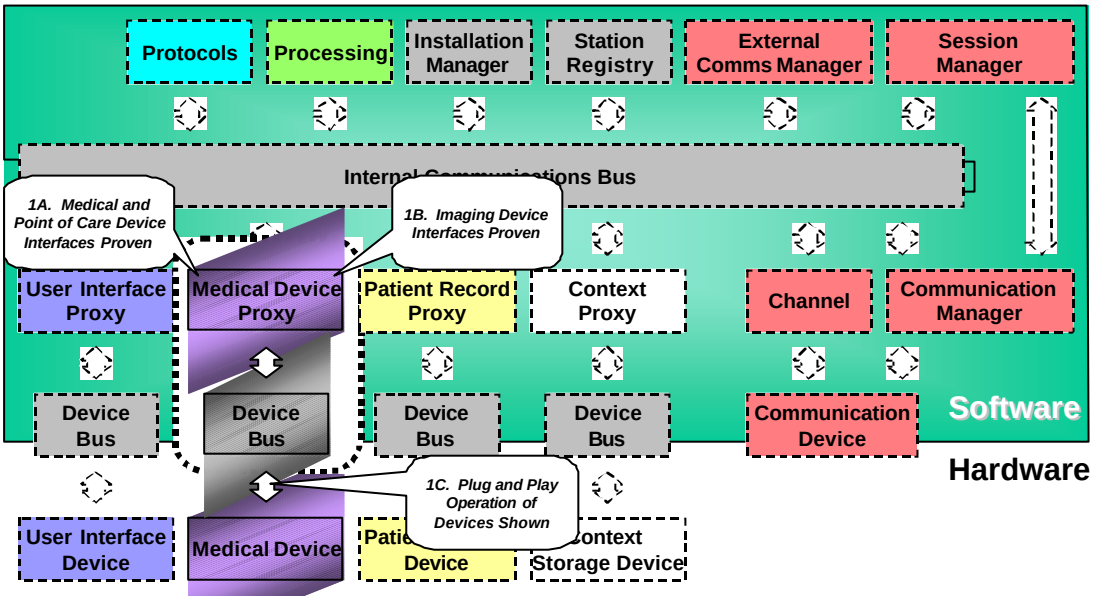


**Figure 81. Elements Addressed in Phase 1**

#### *4.6.1.1 Phase 1A: Standardized Medical Device Interfaces*

**Goal:** Prove out the application interfaces for a representative set of medical devices, excluding imaging devices.

**Approach:**

- This phase shows the operation of a number of medical devices and telemedicine platforms
- Choose candidate platforms to represent the kinds of platforms that are likely to be encountered in near-term implementations. These include traditional telemedicine stations from one or more vendors, a standard PC (desktop or portable), one or more PDAs (having both PalmOS and PocketPC versions would be preferable), and a BCC (bedside monitoring unit) per the work done by the IEEE 1073 committee.
- Choose candidate devices based on kinds of outputs (e.g.,

single scalar value, continuous waveform, etc.) generated and kinds of devices currently found in telemedicine applications, and in light of the availability of applicable standards that can be used as is or as models for the application-level interfaces for these devices.

- Match specific devices with specific platforms.
- Platform developers enlisted for this work implement medical device proxies that implement these interfaces and whatever user interface software is needed to demonstrate the operation of the devices.
- Device manufacturers implement devices capable of at least minimal intelligence (i.e., "self-aware devices" per Figure 53) if not "ideal devices". Ideally, there would be at least two different devices for each device type implemented (e.g., pulse oximeters from two different manufacturers that both implement the same application-level interface). At this point in the process, the devices may (but are assumed not to) be built on device bus technology.
- Evaluate the implementation of these interfaces using tests that pair each platform with its associated devices.

**Product:** A proven set of application-level medical device interfaces

**Rationale:** This step is a necessary foundation for the development of both plug-and-play operation of medical devices and for the leasing and control of medical devices by remote stations (as addressed in Phase 2)

#### 4.6.1.2 Phase 1B: Standardized Imaging Device Interfaces

**Goal:** Demonstrate the application interfaces for a representative set of imaging devices.

**Approach:**

- Enlist a handful of platform developers for implementation of the medical device proxies that will host these interfaces and of any processing and user interface components needed to demonstrate the operation of these devices. Developers should represent a range of platform types (e.g., custom, off-the-shelf PC, PDA, video game box).
- Enlist manufacturers of the selected devices (ideally, at least two for each device type) to implement the application-level interfaces in their devices. At this point in the process, the devices may (but are assumed not to) be built on device bus technology.
- Evaluate the implementation of these interfaces tests that pair platform and device vendors

**Product:** A proven set of application-level medical device interfaces

**Rationale:** This step is a necessary foundation for the development of plug-and-play operation and for leasing and control by remote stations

#### 4.6.1.3 Phase 1C: Plug-and-Play Medical Device Interfaces

**Goal:** Demonstrate plug-and-play operation of the medical and imaging devices.

**Approach:**
- Using the platform and device developers from Phase 1, incorporate selected device bus interfaces (preferably wired and wireless) into the platforms and devices
- As appropriate, share device-specific interface software between platform developers to allow each platform to support multiple devices
- Evaluate the platform and device implementations using a plug-fest approach in which medical devices are swapped at will between each of the platforms

**Product:** A proven set of plug-and-play medical device interfaces

**Rationale:** This step completes the medical device interface work for a limited subset of devices. Given this foundation, the telemedicine community can move forward in several related areas, including development of a richer suite of plug-and-play devices (i.e., expanding the number of devices that telemedicine systems can support) and production of devices that are cheaper and suited for use in a broader range of settings.

### 4.6.2 Phase 2 – Station-to-Station Operation

This phase of the work addresses those aspects of the architecture that relate to basic station-to-station interactions. Figure 82 identifies which elements of the architecture are addressed in this phase. The plan for this phase is:

- Demonstrate the ability to explore the capabilities of a remote station
- Demonstrate the leasing, operating, and vacating of medical device capabilities of a remote station
- Add videoconferencing
- Add other person-to-person communications

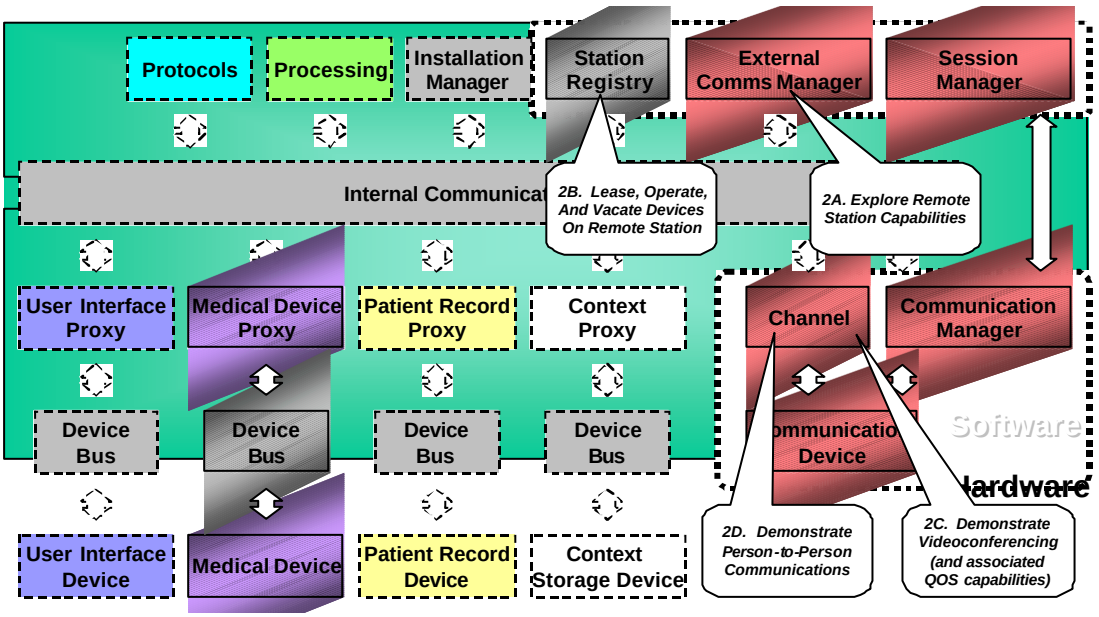


**Figure 82. Elements Addressed in Phase 2**

When complete it should be possible to allow stations from different vendors to explore each other's services, to remotely control each other's medical devices, and to enable remotely located users to communicate both interactively and in store-and-forward fashion.

### 4.6.2.1 Phase 2A: Station Exploration Interfaces

**Goal:** Demonstrate the ability to explore the capabilities of a remote station

**Approach:**
- Using the capabilities developed in Phase 1, incorporate registry capabilities in each platform along with implementations of IP-based channels, the associated communications manager, the external communications manager, and a session manager
- Implement the basic station exploration services needed for medical devices
- As required, implement the additional user interface capabilities needed to support these operations
- Evaluate the implementations using platform-to-platform pairings

**Product:** A proven subset of the registry, channel, communication manager, external communications manager, and station-to-station interfaces

**Rationale:** This step delivers the first instantiations of the key structures needed for station-to-station communications

### 4.6.2.2 Phase 2B: Remote Operation Interfaces

**Goal:** Demonstrate the leasing, operating, and vacating of medical device capabilities of a remote station

**Approach:**
- Building on Phase 2A implement the leasing, and remote operation services needed for medical devices
- As required, implement the additional user interface features needed to support these operations (some of this may be done by sharing of components between collaborating vendors)
- Evaluate the implementations using platform-to-platform pairings

**Product:** A proven set of interfaces for monitoring and control of remote medical and imaging devices

**Rationale:** This step moves us toward vendor independence by making it possible to use any vendor's station to remotely control and monitor medical devices located on any other vendor's station.

### 4.6.2.3 Phase 2C: Videoconferencing Interfaces

**Goal:** Demonstrate the ability to conduct a videoconference with a remote station

**Approach:**
- Using the capabilities developed in Phase 2A, implement channels that support the protocols needed for audio- and video-based communications. Implement telephony-based communications. Implement dynamic bandwidth management so that channel capacity allocation can be changed on user command.
- As required, implement the additional user interface and processing components needed to support these operations
- Evaluate the implementations using platform-to-platform pairings

**Product:** A proven set of videoconferencing and quality of service interfaces

**Rationale:** Current approaches to telemedicine rely heavily on interaction between caregiver and patient or caregiver and caregiver. Delivering the ability to videoconference is key to allowing vendors to recast their systems in a manner compliant with this architecture.

#### 4.6.2.4 Phase 2D: Person-to-Person Communication Interfaces

**Goal:** Demonstrate the ability to support communication between remotely located individuals using means other than audio- and videoconferencing

**Approach:**

- Using the capabilities developed in Phase 2A and 2C, implement channels that support the protocols needed for email, chat, instant messaging, and file sharing.
- As required, implement the additional user interface and processing components needed to support these operations
- Evaluate the implementations using platform-to-platform pairings

**Product:** A proven set of text-/data-focused communication interfaces

**Rationale:** Some approaches to telemedicine rely on text-based messaging and file sharing between caregiver and patient or caregiver and caregiver. Delivering the ability to communicate in these ways (when added to the interactive mechanisms of Phase 2C) provides vendor with the set of tools needed to support virtually any desired approach to human interactions.

### 4.6.3 Phase 3 – Component-based User Interfaces & Processes

This phase of the work formalizes (as standard components) the user interface and processing elements created to date for the station. Figure 83 identifies those elements on which this phase of the architecture concentrates. The plan for this phase is:

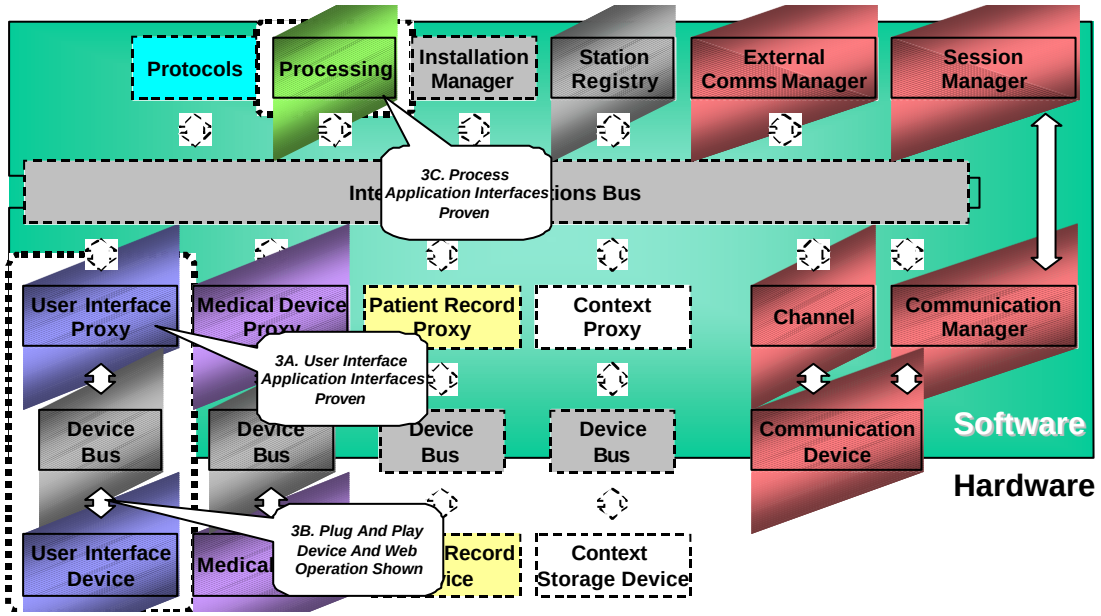


**Figure 83. Elements Addressed in Phase 3**

- Prove out the application interfaces for user interface components created to date
- Demonstrate plug-and-play operation of selected user interface devices and web presentation of user interfaces
- Prove out the application interfaces for processing components created to date

When complete, it will be possible to use external devices (e.g., PDAs) as user interface elements of a station and to swap processing routines that supply the same functionality. In addition, all of the necessary groundwork will have been laid to move towards full component-based station designs.

#### 4.6.3.1 Phase 3A: User Interface Application Interfaces

**Goal:** Prove out the use of protocols and subscription interfaces to handle the interconnection of resources within a station

**Approach:**
- As needed, partition all of the user interfacing functionality built to date into user interface components that present standardized interfaces to the other station components with which they interact.
- Evaluate these "normalized" implementations both in stand-alone and station-to-station modes

**Product:** Proven application-level user interface specifications

**Rationale:** This step lays the groundwork needed for Phase 4 by "cleaning up" the user interface and processing components created so far

#### 4.6.3.2 Phase 3B: User Interface Devices

**Goal:** Demonstrate plug-and-play operation of selected user interface devices and web presentation of user interface.

**Approach:**
- Building on the work of Phase 1C, incorporate the ability for selected interface devices (e.g., PDAs) to present themselves to the platform via the device bus interfaces selected for the platform
- Evaluate the platform and device implementations using a plug-fest approach in which medical devices are swapped at will between each of the platforms

**Product:** A proven set of plug-and-play user interface device interfaces

**Rationale:** This step, when coupled with Phase 3A, makes it possible for a user to move from station to station and to dynamically access the station's capabilities.

#### 4.6.3.3 Phase 3C: Processing Application Interfaces

**Goal:** Prove out the application interfaces for processing components created to date

**Approach:**
- As needed, partition all of the processing functionality built to date into processing components that present standardized interfaces to the other station components with which they interact.
- Evaluate these "normalized" implementations both in stand-alone and station-to-station modes

**Product:** Proven application-level user interface and processing interface specifications

**Rationale:** This step lays the groundwork needed for Phase 6 by "cleaning up" the user

interface and processing components created so far

### 4.6.4 Phase 4 – Component-based Stations

This phase of the work addresses those aspects of the architecture that relate to basic station-to-station interactions. Figure 84 identifies which elements of the architecture are addressed by this phase. The plan for this phase is:

- Demonstrate operation of internal communications bus
- Prove out the use of protocols and subscription interfaces to handle the interconnection of resources within a station
- Demonstrate a distributed station implementation in which station components are capable of announcing their presence and being added dynamically to the station under the control of protocols

When complete it should be possible to build stations from distributed collections of station components developed by different vendors.

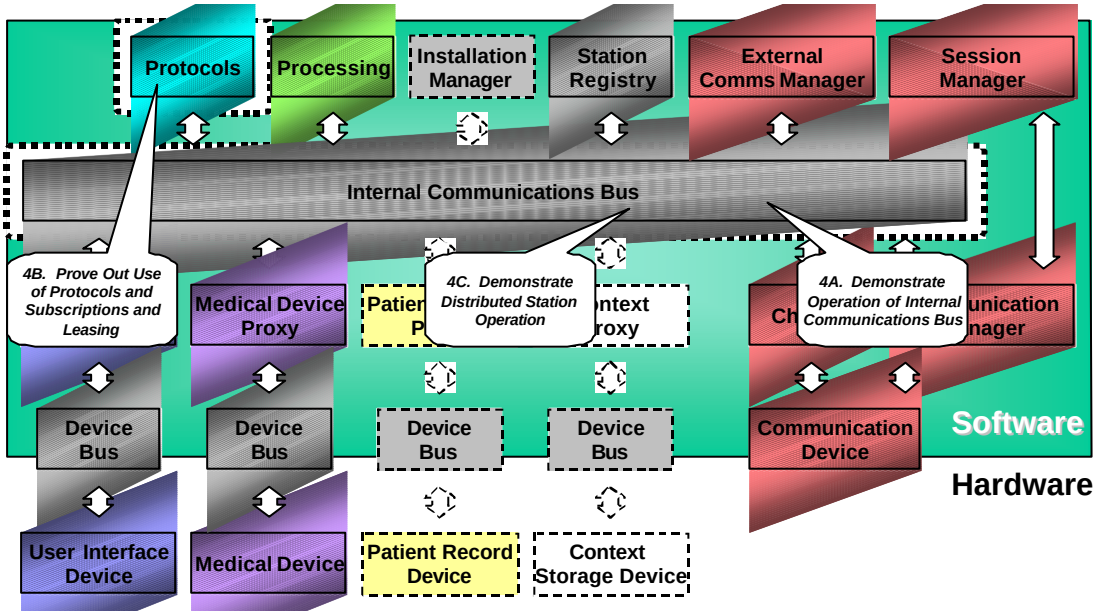


**Figure 84. Elements Addressed in Phase 4**

#### *4.6.4.1 Phase 4A: Internal Communications Bus*

**Goal:** Demonstrate operation of internal communications bus

**Approach:**

- Render the bus manager, the media manager for a selected IP-based medium, the administrative channel, and synchronous and asynchronous channel capabilities.
- Rework existing station internals to make use of these bus capabilities
- As required, implement the additional user interface capabilities needed to support these operations
- Evaluate the implementations using platform-to-platform pairings

**Product:** A proven set of plug-and-play medical device interfaces

**Rationale:** The internal communications bus is the necessary foundation for

distribution of station components and the creation of virtual stations. This step reworks proprietary internal communication approaches by allowing the architecture-compliant components developed in the earlier phases to interact with one another over the internal communications bus. At the end of this step, interconnections will be "hard-wired" only to the extent that protocols have not yet been implemented to enable for dynamic association through leasing and subscription; however, registration will be enabled (i.e., a medical instrument or user interface attaching to the station will communicate with the registry over the internal communications bus to allow the device to be registered with the station.

#### 4.6.4.2 Phase 4B: Protocols and Subscription Interfaces

**Goal:** Prove out the use of protocols and subscription interfaces to handle the interconnection of resources within a station

**Approach:**
- As needed, extend all of the components built to date (registry, communications manager, session manager, channels, medical device proxies, processing components, and user interface components) to ensure that they support the registration and subscription model.
- Create protocols that specify how components are to subscribe to one another in support of the various operational capabilities implemented to date
- As required, implement the additional user interface capabilities needed to support these operations
- Evaluate the implementations both in stand-alone and station-to-station modes

**Product:** Proven protocol and subscriptions interface specifications

**Rationale:** Given solid starts on the two major external interfaces for current telemedicine systems (station-to-station communications and station-to-device communications), the goal is to now (before adding any other capabilities to the station) lay a foundation for the internal interfaces that allow for distributed, component-based design of stations. The motivation for doing this at this point in the evolution of the architecture is to strike a balance between two key needs. One the one hand, while we could enforce these interfaces from the very start, a goal of the previous phases is to allow vendors with current products to slowly morph those products to a point where they are able to support medical devices from any vendor and can interact with stations from any other vendor. One the other hand, while we could wait later in the evolution to address the station-internal interfaces, it might make the task of retrofitting components overwhelming.

#### 4.6.4.3 Phase 4C: Distributed Plug-and-Play Station Operation

**Goal:** Demonstrate a distributed station implementation in which station components are capable of announcing their presence and being added dynamically to the station under the control of protocols

**Approach:**
- Using the capabilities developed so far, distribute the components across a number of special purpose platforms (e.g., a home gateway that

hosts the registry, the channels and associated communications components; “access points” that host medical devices; networked computers that host protocols and processes; and various user interface devices) around a local network. To demonstrate the power of dynamic protocols, multiple instances of each kind of resource should be included in this network..

- Building of Phase 4A, implement the media manager for a second internal communications bus medium.
- Implement any internal communications bus interfaces that might be lacking in order to allow over-the-network (i.e., station-internal) interactions between components residing on separate media on the internal bus.
- Increase the intelligence of the existing protocols to handle dynamic selection of resources
- As required, implement the additional user interface needed to support these operations
- Evaluate the distributed implementation both as stand-alone and station-to-station modes.

**Product:** A proven set of station-internal component communication interfaces, mechanisms for handling heterogeneous internal communications media, and the ability to select among multiple devices of a given type available on the bus.

**Rationale:** This step is foundational to establishing component-based telemedicine stations that can be assembled in mix-and-match fashion using components from different component vendors (telemedicine and otherwise).

### 4.6.5 Phase 5 – Patient Records

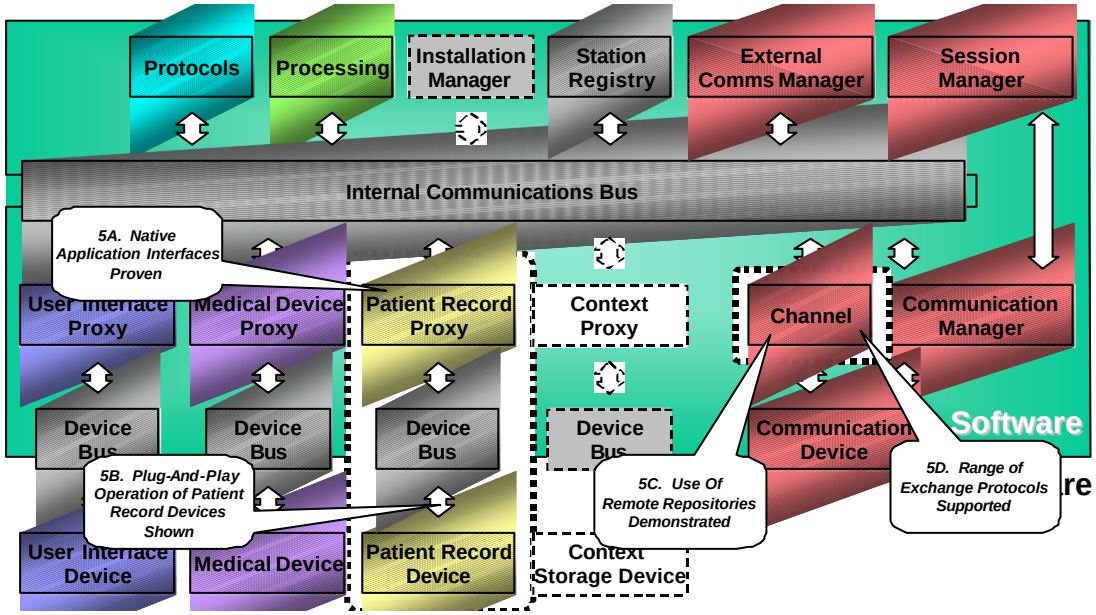


**Figure 85. Elements Addressed in Phase 5**

This phase of the work addresses those aspects of the architecture that relate to the local management of patient records and the exchange of these records between remote stations. Figure 85 identifies the elements on which this phase of the architecture concentrates. The plan for this phase is:

- Prove out the native application-level interfaces for the patient record repository
- Demonstrate plug-and-play operation of patient record devices
- Demonstrate the ability to employ patient record services provided by one or more remote stations
- Demonstrate the ability to negotiate and support a range of patient record exchange protocols (e.g., COAS, HL7, DICOM)

When complete it should be possible to read and write patient record cards, to allow a station to access a repository anywhere on the telemedicine network, to treat multiple repositories as a single virtual repository, and to allow records to be exchanged with remotely located record servers employed in hospitals, clinics, etc.

### 4.6.5.1 Phase 5A: Native Patient Record Interfaces

**Goal:** Prove out the native application-level interfaces for the patient record repository

**Approach:**
- For each of the platforms used to date, build the repository infrastructure and application level interfaces for each of the medical and imaging devices addressed to date
- Build the application level interfaces needed to handle text and audio record entries generated by users
- Extend protocols to make them patient record-aware
- Implement the additional user interface elements needed to support navigation and review of a repository and its records
- Evaluate the implementations on each platform

**Product:** A proven "native" application-level patient record interface

**Rationale:** Patient records represent the final major capability seen in current systems. Implementing this capability gives vendors the wherewithal to render their existing capabilities in a way that is fully architecture-compliant.

### 4.6.5.2 Phase 5B: Patient Record Devices

**Goal:** Demonstrate plug-and-play operation of patient record devices.

**Approach:**
- Building on the work of Phase 1C and 3B, incorporate the ability for each class of patient record devices to be added in plug-and-play fashion to whatever device buses have been selected for each type of platform
- For both these pluggable devices and the fixed repository devices (used in Phase 5A), implement the interfaces related to exploring and managing these devices (e.g., determining device characteristics, acquiring or releasing storage space, etc.)
- Extend the existing protocols to make use of the services provided by these new interfaces
- Implement the additional user interface elements needed to support these services

- Evaluate the record media implementations using a plug-fest approach where appropriate

**Product:** A proven set of plug-and-play patient record device interfaces and interfaces related to media

**Rationale:** This step allows for the integration of devices, such as the Army's Personal Information Carrier, into telemedicine systems and for the incorporation of intelligence into protocols such that they can decide whether a user-specified operation is viable and, if not, what acceptable alternatives would be.

#### 4.6.5.3 Phase 5C: Virtual Record Operations

**Goal:** Demonstrate the ability to employ patient record services provided by one or more remote stations

**Approach:**
- Implement the interfaces needed to integrate multiple local repositories leased by a station into a single virtual repository
- Implement the interfaces needed to integrate multiple distributed repositories (i.e., both local and/or remote) leased by a station into a single virtual repository (the assumption here is that accessing remote repositories will entail implementing a new channel for external communications that supports the native repository interfaces of Phases 5A and 5B).
- Extend the protocols to make use of these virtual repository interfaces,
- Implement the additional user interface capabilities needed to support virtual repository operations
- Evaluate the implementations by allowing each platform implemented to date to access the repositories associated with any or all of the other platforms.

**Product:** A proven distributed patient record interface

**Rationale:** This step moves us toward vendor independence by making it possible to use any vendor's station to remotely access a patient's records even if they are scattered across repositories on a number other compliant telemedicine stations.

#### 4.6.5.4 Phase 5D: Patient Record Protocol Negotiation

**Goal:** Demonstrate the ability to negotiate and support a range of patient record exchange protocols (e.g., COAS, HL7, DICOM)

**Approach:**
- For each platform used to date, implement external communication channels that support the protocols needed for interaction with external record servers and give each of these channels the ability to map between external protocols constructs and the architecture's "native" repository constructs.
- Implement the protocol required for exploring the patient record exchange protocols supported by a remote station and for allowing two stations to negotiate which mutually-supported protocol(s) will be used in a given session.
- As required, extend the patient record-related portions of the user

interface to accommodate navigation and review of remotely-located non-native repositories

- For each platform, evaluate the ability to access remote, non-native repositories

**Product:** A proven set of interfaces for interoperation with devices supporting other patient record data exchange standards

**Rationale:** This completes the patient record needs by making it possible for a telemedicine station to access not only repositories stored on other architecture-compliant telemedicine stations but also those on other industry standard record servers.

### 4.6.6 Phase 6 -- Security

This phase of the work addresses those aspects of the architecture that relate to basic station-to-station interactions. Figure 86 identifies the elements that are the focus of this phase the architecture's evolution. The plan for this phase is:

- Demonstrate the ability to identify and authenticate a user attempting to access a station either locally or remotely and to selectively grant or deny access to services and data based on this information
- Demonstrate the ability to negotiate and secure station-to-station communications and to negotiate and support station-to-station quality of service
- Demonstrate the ability to negotiate and secure station-internal communications and to negotiate and support station-internal quality of service

When complete it should be possible to allow stations from different vendors to explore each other's services and to remotely control each other's devices.

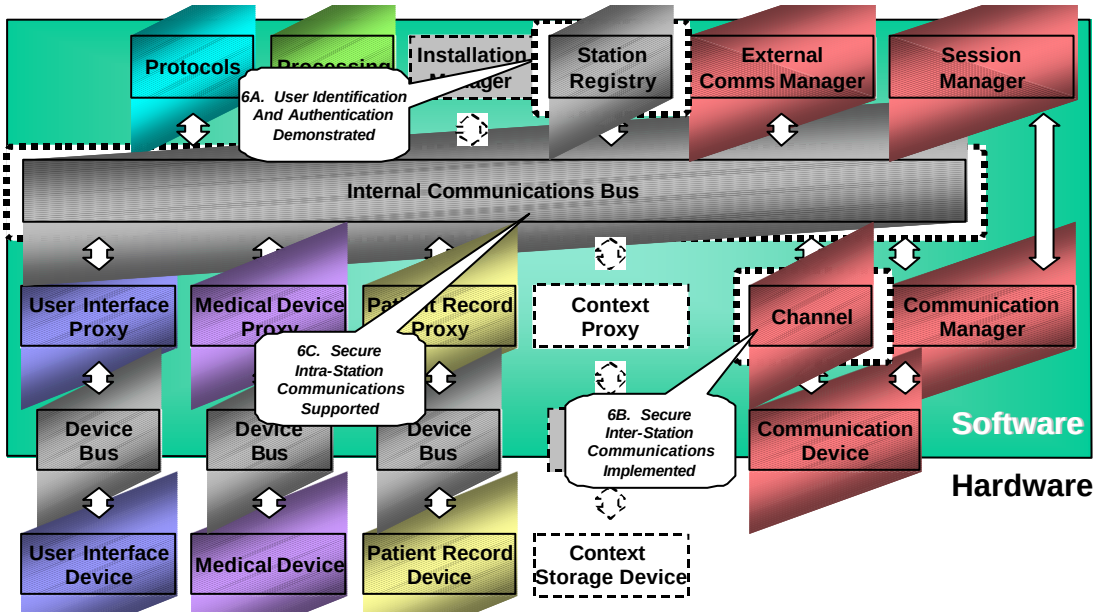


**Figure 86. Elements Addressed in Phase 6**

#### 4.6.6.1 Phase 6A: User Authentication and Access Control

**Goal:** Demonstrate the ability to identify and authenticate a user attempting to access a station either locally or remotely and to selectively grant or deny access to services and data based on this information

**Approach:**
- Retrofit existing platforms with a range of mechanisms that support user identification and authentication (e.g., passwords, tokens, biometrics)
- Implement the access control decision mechanisms needed by a station to determine with data and services on that station can be accessed by a given user
- Implement the security extensions needed in the infrastructure for each station component to enforce access control decisions
- Implement the additional user interface features needed to support these operations
- Evaluate the implementations with both local and remote users attempting to log in and use controlled station services

**Product:** A proven set of user I&A and access control interfaces

**Rationale:** This is a needed building block if vendors are to move from closed systems characteristic of today's telemedicine networks to open systems.

#### 4.6.6.2 Phase 6B: Communications Security and QOS

**Goal:** Demonstrate the ability to negotiate and secure station-to-station communications and to negotiate and support station-to-station quality of service

**Approach:**
- To the extent that this has not already been done, for each of the protocols implemented so far that support station-to-station communications, implement on each platform the mechanisms needed to secure these protocols
- Implement the protocol required for exploring the security mechanisms supported by a remote station and for allowing two stations to negotiate which mutually supported mechanisms will be used in a given session.
- Implement any additional required QOS protocols not implemented in Phase 2.
- As required, implement the additional user interface elements needed to support these operations
- Evaluate the implementations using platform-to-platform pairings

**Product:** A proven set of protocols for securing station-to-station communications

**Rationale:** This is a needed building block if vendors are to move from closed systems characteristic of today's telemedicine networks to open systems.

#### 4.6.6.3 Phase 6C: Station Security

**Goal:** Demonstrate the ability to negotiate and secure station-internal communications and to negotiate and support station-internal quality of service

**Approach:**
- Implement on each platform the mechanisms needed to secure internal communications bus communications

- Implement the protocol required for exploring the security mechanisms supported by station components and for allowing station components to negotiate which mutually supported mechanisms will be used in a given session.
- Extend protocols with these capabilities.
- Implement any additional required QOS protocols not implemented in Phase 4.
- As required, implement the additional user interface elements needed to support these operations
- Evaluate the implementations by creating systems from the various distributed platform elements.

**Product:** A proven set of protocols for securing station-to-station communications.

**Rationale:** This step delivers the final building block required for securing operation of distributed stations and systems.

### 4.6.7 Phase 7 – Dynamic Configuration

This phase of the work addresses those aspects of the architecture that relate to basic station-to-station interactions. Figure 87 identifies which elements of the architecture this phase addresses. The plan for this phase is:

- Demonstrate the ability to temporarily and permanently retrieve and install components needed to support the operation of other components on a station
- Demonstrate the ability to dynamically configure a station's operation based on patient and caregiver context data

When complete it should be possible to allow stations from different vendors to explore each other's services and to remotely control each other's devices.

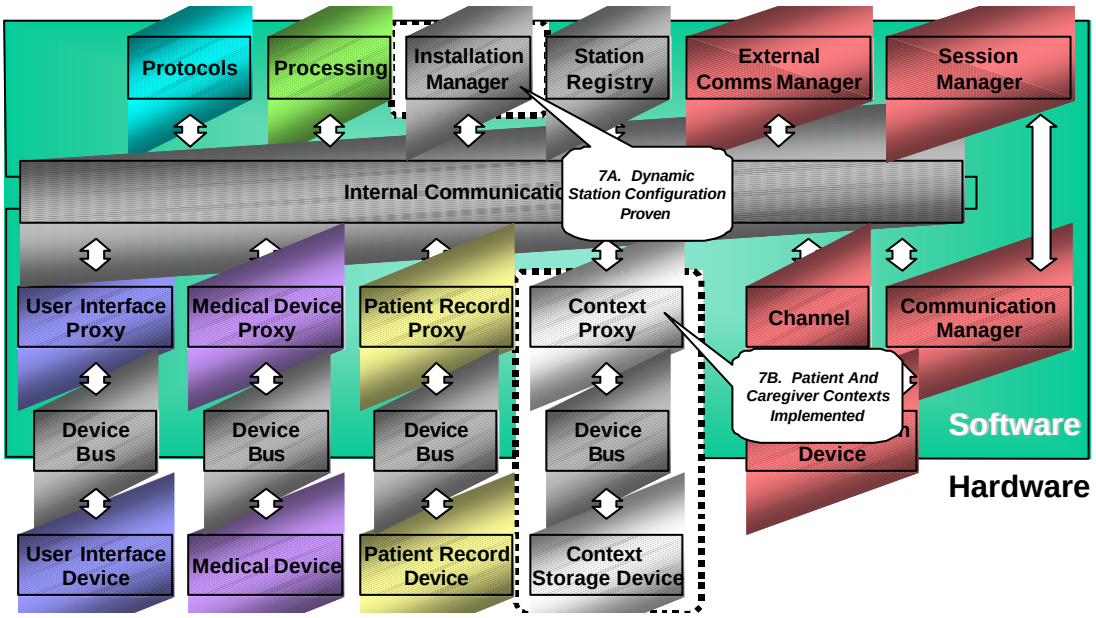


**Figure 87. Elements Addressed in Phase 7**

### 4.6.7.1 Phase 7A: Dynamic Station Configuration

**Goal:** Demonstrate the ability to temporarily and permanently retrieve and install components needed to support the operation of other components on a station

**Approach:**
- For each of the platforms implement the protocol needed to identify to a vendor server the software packages required by the station and to download these packages from the station
- For each platform, implement an installation manager capable of verifying the authenticity and integrity of a software package and of installing on the station the components it contains
- Implement a vendor server station and install on the station packages for each of the platforms that will be used when each kind of pluggable device addressed to date in the development of this architecture
- As required, implement the additional user interface capabilities needed to support these operations
- Evaluate the implementations on each platform by clearing the platforms of all knowledge of the pluggable devices and then plugging each device into each platform and having the platform retrieve the appropriate software packages from the vendor server

**Product:** A proven set of interfaces for requesting, downloading, and dynamically installing software packages from vendor servers.

**Rationale:** This step makes it possible for vendors to create systems that require little more of station users than plugging a base station into external communications and then plugging devices into the base station.

### 4.6.7.2 Phase 7B: Contexts

**Goal:** Demonstrate the ability to dynamically configure a station's operation based on patient and caregiver context data

**Approach:**
- For each platform type, implement the default context capability
- If required, extend all applicable protocols developed to date to support leasing and subscription by a context
- For each station, implement the processing of patient and caregiver contexts
- Incorporate the ability for context storage devices to present themselves to the platform via the device bus interfaces selected for each platform
- As required, implement the additional user interface components needed to support these operations
- Evaluate the implementations by editing each platform's default context and proving that it bootstraps itself into a base configuration consistent with the context. Next introduce patient and caregiver contexts to each platform by each mechanism supported by each platform and prove that the station properly reconfigures itself in accordance with these contexts

**Product:** A proven subset of the registry, channel, communication manager, and station-to-station interfaces

**Rationale:** This step allows vendors to implement user-specific dynamic customization of stations.

### 4.6.8 Phase 8 – System Administration

This phase of the work addresses those aspects of the architecture that relate to basic station-to-station interactions. Figure 88 identifies the architecture elements addressed in this phase. The plan for this phase is:

- Demonstrate the ability of a station to add itself to or remove itself from a registry
- Demonstrate the use a network of registry servers to provide the ability to locate stations by name and by station attributes

When complete it should be possible to allow independent stations to locate one another without knowing specific addresses in advance.

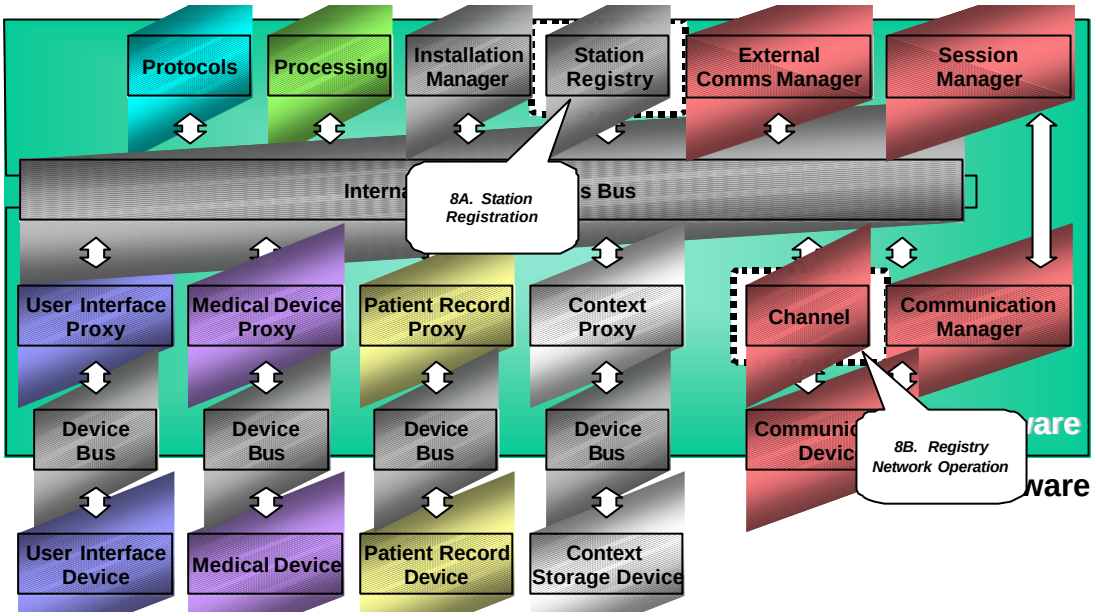


**Figure 88. Elements Addressed in Phase 8**

#### *4.6.8.1 Phase 8A: Station Registration*

**Goal:** Demonstrate the ability of a station to add itself to or remove itself from a registry

**Approach:**
- For each of the platforms, implement the protocol needed to allow a station to register itself with a registry server
- Implement a registry server
- Implement the additional user interface capabilities needed to support these operations
- Evaluate the implementations on each platform by registering each station with the server

**Product:** A proven set of registry service interfaces

**Rationale:** This is the final step in making stations fully plug-and-play.

#### *4.6.8.2 Phase 8B: Registry Network Operations*

**Goal:** Demonstrate the use a network of registry servers to provide the ability to locate stations by name and by station attributes

**Approach:**
- Extend the registry server to allow for operation as part of a server network
- Deploy a limited network of registry servers
- Evaluate the implementations on each platform by registering different stations both with a common server and with different servers and using each station's new interface capabilities to search for the other stations by name and by attributes

**Product:** A proven registry server design

**Rationale:** A distributed registry server network will be foundational to the creation of open telemedicine networks. This step provides the architectural design needed for the creation of servers that will populate this network.

### 4.6.9 Phase 9 – Multi-station Conferencing

This phase of the work addresses operations in which more than two stations are involved in a given session. Figure 89 identifies on which elements of the architecture this phase concentrates. When complete it should be possible to dynamically add and delete multiple stations to and from a session.

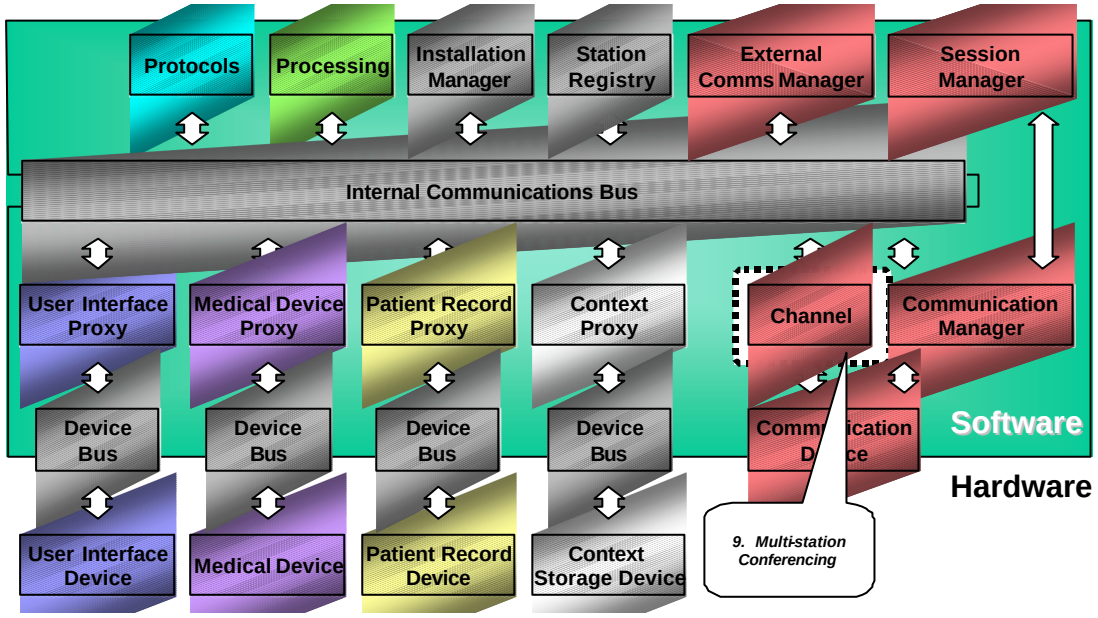


**Figure 89. Elements Addressed in Phase 9**

**Goal:** Demonstrate the ability to operate in a multi-station (i.e., 3 or more simultaneously) mode and to dynamically add and remove stations from multi-station conferences

**Approach:**
- For each of the platforms, extend the relevant channel specifications to include the ability to handle conferencing in which the stations involved in the conference decide among themselves which station will act as host for the conference (the point through which all communications are routed)

- Evaluate through different mixes of platforms with different stations being dynamically added and removed from conferences

**Product:** A proven set of protocols for use in conference-oriented operations

**Rationale:** This step enables vendors to move beyond standard two-party operations.

## 4.7 Organizing to Reach The Goal

As has been said a number of times throughout this document, if this effort is ever to mean anything, it will be because the telemedicine community has decided to come together for the purpose establishing a common approach to technical interoperability. That the ideas in the document are advanced is far less important than whatever ideas are ultimately agreed upon enjoy a broad base of support from both vendors and users of systems and components. In the end, this is likely to occur only if the telemedicine community organizes itself for this purpose.

One structure for doing this is shown in Figure 90. The heart of the organization consists of a management council that exists to establish and manage needed relationships with external organizations, review progress of the technical teams, and set priorities for these teams as the overall effort is carried forward, and the system engineering team that oversees all technical development work.

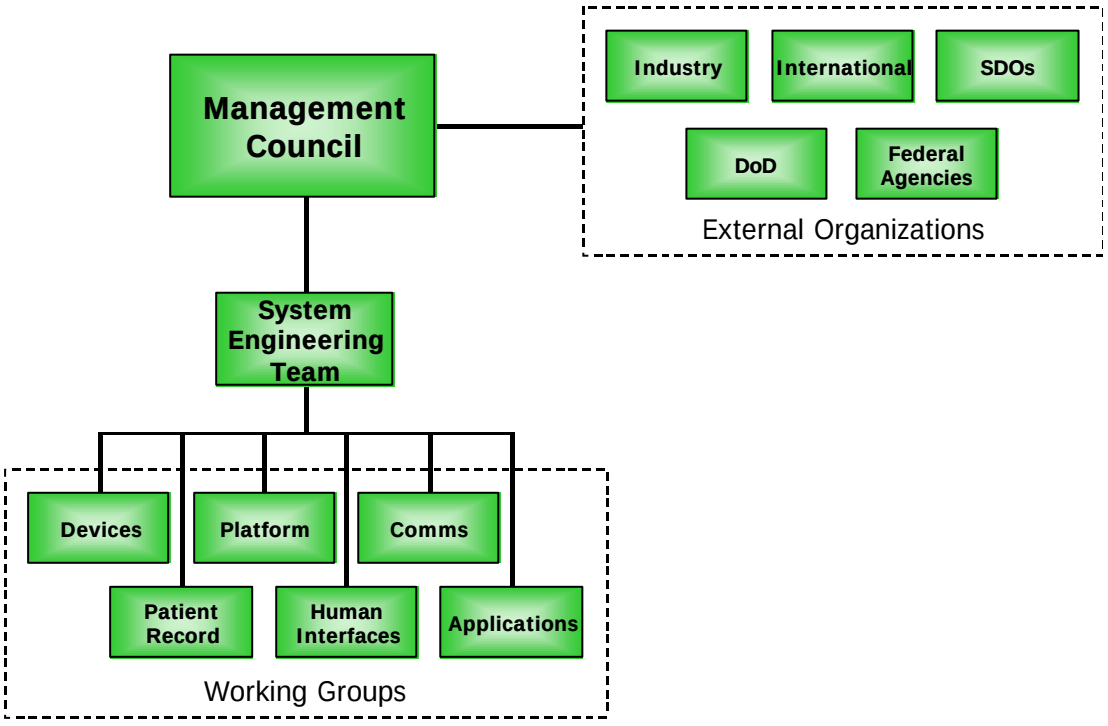


**Figure 90. A Suggested Organizational Structure**

The technical work of the team is divided among the seven groups shown. Management of the overall architectural framework and resolution of interface issues between teams is

the responsibility of the system engineering team. The working group on devices worries about interface specifications for all devices to be supported telemedicine systems and about the device bus specifications. The communications working group handles both internal communications bus issues and standardization of external communications interfaces. The platform working group handles the standardization issues related to the registry, contexts, the installation manager, registry servers, and vendor servers, and all security services not related to communications (including the audit log), and addresses questions related to underlying hardware and software environments on which the stations might be built. The patient record group addresses standardization of the medical repository services and the internal structure of the records that they contain. This group also has primary responsibility for all terminology and encoding issues related to the station. The human interfaces group defines all of the service interfaces that human interface components present to the rest of the components in a station and addresses how various interface devices are to be handled by the architecture. The applications working group is responsible for standardizing interfaces for the various kinds of software processing modules (e.g., statistical, medical logic) than might be employed within the context of a telemedicine station. This group also has responsibility for standardizing the structure of protocols and services associated with their processing.

## 4.8 Finding a Home for The Specifications

Once the job of hammering out an initial set of community-accepted specifications has been accomplished, it will be important that a long-term home be found for the specifications. As system developers and users embrace the specifications, it is inevitable that problems will be found that did not surface in the earlier work and that extensions to the specification will be requested. While the telemedicine community might want to form its own group for managing these sorts of issues, a better alternative is to try to find a home for the specification in a well established "standards development organization" that already has in place the organizational structure and procedures needed to manage the evolution of specifications.

# 5 Final Thoughts

As Bauer and Ringel have said, it is very likely that "telemedicine will ultimately revolutionize healthcare -- restructuring virtually every relationship and activity that define late twentieth century medicine." As the Internet becomes more pervasive, nano-technology allows us to create diminutive devices that are just as capable as their hospital-based predecessors, and intelligent software makes specialized clinical knowledge accessible to less skilled clinicians and even patients, change in our care delivery structures and practices is inevitable.

At the same time, this change won't come easily or even quickly. In their article, "Will Disruptive Innovations Cure Health Care?" (Harvard Business Review, September-October 2000, pp 102-112), Clayton Christensen, Richard Bohmer, and John Kenagy write:

> "…the vast majority of research funding from the National Institutes of Health is aimed at learning to cure diseases that historically have been incurable. Much less is being spent on learning how to provide the health care that most of us need most of the time in a way that is simpler, more convenient, and less costly."

Change won't come quickly or easily because is not our current priority. Yet, as they note later in their article, it is at exactly at this point – the reengineering of care delivery structures – that our nation is likely to realize the greatest improvements in health care.

In order for change to be effected a number of technology-related barriers must be overcome, including the lack of a rich suite of devices suitable for use in non-hospital settings, the high cost of current telemedicine systems, and the inability to communicate universally between all telemedicine stations. While not a silver bullet that addresses all of these needs, the existence of generally accepted interoperability specifications is foundational to overcoming these barriers and others. Without these sorts of specifications, system integration will continue to be labor-intensive and, therefore, the cost of systems will remain high and it will be difficult to develop a mass market for these sorts of systems. Without industry-accepted interoperability specifications, we will not see the development of a robust device market that delivers to us both a wider variety of device types and lower cost devices. Finally, without these specifications, the whole telemedicine equipment market itself will be slower to develop as customers, unable to simply buy off the shelf and have their systems work with those owned by other organizations, are reluctant to commit to one vendor's solution over another's.

While some parts of the specifications proposed in this document are based on ideas proven out in prototype systems, some parts are not and need to be evaluated. As such, this document is simply a starting point for a larger effort that must be pursued by the telemedicine community, and this is as it should be. In as much as the point of this specification is interoperability, it will only succeed as it is proven trustworthy and as it gains a base of broad support within the telemedicine community. Even if the final specification looks nothing like what is proposed in this initial strawman, as long as the final specification is embraced by the community at large, the telemedicine community will have achieved its goal in this key area.