

Kansas Information Technology Executive Council Security Council

Logging Best Practices

Whitepaper 04-10-2008

This Whitepaper on logging and audit logs is a Kansas ITEC Security Council best practices document meant to inform and describe to IT workers in various disciplines the nature of logging in its various forms. This paper is also meant to describe the different types of Log files that can be used for security related investigations and how these logs used collectively can tie activities together to establish an irrefutable chain of events leading up to an event. It is not only considered a best practice to monitor and retain log information; it may be required by law for certain organizations. The Federal Information Security Management Act of 2002 (FISMA), the Health Insurance Portability and Accountability Act of 1996 (HIPAA), the Sarbanes-Oxley Act of 2002 (SOX), the Gramm-Leach-Bliley Act (GLBA), and the Payment Card Industry Data Security Standard (PCI DSS) all require organizations to acquire and maintain log data to varying degrees. Organizations will inevitably have differing business models, and by definition have different logging requirements, but all should have a mandatory policy regarding the practice of capturing, storing, archiving and monitoring of log data from devices deemed critical to the infrastructure, as well as a minimum standards policy regarding log information generated by less critical devices.

A log is a record of the events occurring within an organization's systems and networks. Logs are composed of log entries; each entry contains information related to a specific event that has occurred within a system or network. There are many "LOG" files available to most system administrators, database analysts, application developers, and security analysts that can be used to understand and document a set of activities that may have security related implications. This coupling of related LOG files with inter-related transactions can be used when understood and designed correctly as a virtual security process and trail for IT Security Administration. These different log files can reside across the standard OSI Model framework. Many of these files are automatically set up upon installation of both data networking products but also database systems, and application event monitors.

There is a different use for each of these LOG files. They are established separately when designed appropriately so that not all good, bad, and error transactions all end up in the same LOG file structure. This causes problems when a system administrator or security analyst is trying to work through millions of rows of a large log file that contains good transactions (which is really a history file of appropriate predetermined logged transactions) bad transactions (error files that contain bad transactions such as application layer transactions that do not complete appropriately or are intentionally designed to not process do to business or data edit rules), and security logs which contain transactions that appear to have security implications (such as attempted logins or scanning). Each log file that is used must be managed for appropriate size, retention, backup, and archiving when necessary to remove and clean the log files on a regular basis for temporary or permanent backup and data access and reporting. Provisions should be made in specific OSI layers for dealing with LOG files that fill up available DASD so they do not begin to write on top of themselves or overlay transactions. There may be additional steps taken when files are full that stop operations at the database, application, or operating system levels.

Many organizations' have legacy application layer systems in their organizations. These organizations believe many times that unless a large legacy application has logging functions already built in that they cannot go back and add in these functions without great effort and expense. This is not the case. A major use of this document is to understand that this capability can be augmented especially in legacy files with other logging functions and implemented at the application layers with minimal updating in the application system itself. Think of a transaction LOG file in the application layer as just another output data set to write transactions to. Do an analysis on the application and it's most critical and vulnerable business functions (creating or modifying a driver's license or abating penalty or interest on a tax account). These most vulnerable functions in the application layer are the areas programming staff need to insert to make a snapshot of the transaction when it happens and capture all the pertinent LOG file information and write the record. Then the new transaction records can be combined in a single good transaction log file or broken out if specific and separate LOG files are needed for these new transaction sets. When this remediation is made in these systems and the new LOG files are populated remember to manage the files for size, space, and usage as required by all LOG files and stated above.

Capturing Log Data

Security Software

Most organizations have deployed several different types of software designed to enforce security policy, detect malicious activity, protect systems and data, and aid in incident response. These software solutions by their very nature are a major source of valuable log data. Adapted from NIST SP800-92, the following list is an example of a few of the many network and host based security software log sources:

- **Antimalware Software.** The most common form of antimalware software is antivirus software, which typically records all instances of detected malware, file and system disinfection attempts, and file quarantines. Additionally, antivirus software might also record when malware scans were performed and when antivirus signature or software updates occurred. Antispyware software and other types of antimalware software (e.g., rootkit detectors) are also common sources of security information.
- **Intrusion Detection and Intrusion Prevention Systems.** Intrusion detection and intrusion prevention systems record detailed information on suspicious behavior and detected attacks, as well as any actions intrusion prevention systems performed to stop malicious activity in progress. Some intrusion detection systems, such as file integrity checking software, run periodically instead of continuously, so they generate log entries in batches instead of on an ongoing basis.
- **Remote Access Software.** Remote access is often granted and secured through virtual private networking (VPN). VPN systems typically log successful and failed login attempts, as well as the dates and times, each user connected and disconnected, and the amount of data sent and received in each user session. VPN systems that support granular

access control, such as many Secure Sockets Layer (SSL) VPNs, may log detailed information about the use of resources.

- **Web Proxies.** Web proxies are intermediate hosts through which Web sites are accessed. Web proxies make Web page requests on behalf of users, and they cache copies of retrieved Web pages to make additional accesses to those pages more efficient. Web proxies can also be used to restrict Web access and to add a layer of protection between Web clients and Web servers. Web proxies often keep a record of all URLs accessed through them.
- **Vulnerability Management Software.** Vulnerability management software, which includes patch management software and vulnerability assessment software, typically logs the patch installation history and vulnerability status of each host, which includes known vulnerabilities and missing software updates. Vulnerability management software may also record additional information about hosts' configurations. Vulnerability management software typically runs occasionally, not continuously, and is likely to generate large batches of log entries.
- **Authentication Servers.** Authentication servers, including directory servers and single sign-on servers, typically log each authentication attempt, including its origin, username, success or failure, and date and time.
- **Firewalls.** Like routers, firewalls permit or block activity based on a policy; however, firewalls use much more sophisticated methods to examine network traffic. Firewalls can also track the state of network traffic and perform content inspection. Firewalls tend to have policies that are more complex and generate more detailed logs of activity than routers.
- **Network Quarantine Servers.** Some organizations check each remote host's security posture before allowing it to join the network. This is often done through a network quarantine server and agents placed on each host. Hosts that do not respond to the server's checks or that fail the checks are quarantined on a separate virtual local area network (VLAN) segment. Network quarantine servers log information about the status of checks, including which hosts were quarantined and for what reasons. Network Quarantine Servers are most commonly associated and used in conjunction with Network Access Control (NAC) systems, as well as 802.1x port authentication.

Hardware

- **Network Switches.** While switches are generally simple devices, they can provide some minute device level information that can be helpful in troubleshooting connectivity problems as well as bandwidth monitoring, port performance and other low level information about the devices that it is servicing.
- **Routers.** Routers may be configured to permit or block certain types of network traffic based on a policy. Routers that block traffic are usually configured to log only the most basic characteristics of blocked activity. Routers used for filtering traffic are commonly referred to as "packet filters". Routers used in place of a traditional firewall are not

considered a strong security solution as they fall short in many critical areas such as logging, packet state tracking and their inherent vulnerability to Denial of Service (DoS) attacks.

Applications

Most Commercial of-the-shelf (COTS) applications have some sort of logging mechanism that is activated by default, although the type of logging and methods may vary significantly. Minimum logging level should include application specific event data as well as security related data. All entries should have adequate time and date information including month, day, year, hour, minutes, and seconds. Any custom software packages developed in house or purchased should adhere to the same standards as well as any future software development. NIST SP800-92 Section 2.1.3 lists several common types of log data as well as potential benefits for each:

- **Client requests and server responses**, which can be very helpful in reconstructing sequences of events and determining their apparent outcome. If the application logs successful user authentications, it is usually possible to determine which user made each request. Some applications can perform highly detailed logging, such as e-mail servers recording the sender, recipients, subject name, and attachment names for each e-mail; Web servers recording each URL requested and the type of response provided by the server; and business applications recording which financial records were accessed by each user. This information can be used to identify or investigate incidents and to monitor application usage for compliance and auditing purposes.
- **Account information** such as successful and failed authentication attempts, account changes (e.g., account creation and deletion, account privilege assignment), and use of privileges. In addition to identifying security events such as brute force password guessing and escalation of privileges, it can be used to identify who has used the application and when each person has used it.
- **Usage information** such as the number of transactions occurring in a certain period (e.g., minute, hour) and the size of transactions (e.g., e-mail message size, file transfer size). This can be useful for certain types of security monitoring (e.g., a ten-fold increase in e-mail activity might indicate a new e-mail-borne malware threat; an unusually large outbound e-mail message might indicate inappropriate release of information).
- **Significant operational actions** such as application startup and shutdown, application failures, and major application configuration changes. This can be used to identify security compromises and operational failures.

Operating Systems

Operating systems (OS) for servers, workstations, and networking devices (e.g., routers, switches) usually log a variety of information related to security. The most common types of security-related OS data are as follows:

- **System Events.** System events are operational actions performed by OS components, such as shutting down the system or starting a service. Typically, failed events and the most significant successful events are logged, but many OSs permit administrators to specify which types of events will be logged. The details logged for each event also vary widely; each event is usually time stamped, and other supporting information could include event, status, and error codes; service name; and user or system account associated with an event.
- **Audit Records.** Audit records contain security event information such as successful and failed authentication attempts, file accesses, security policy changes, account changes (e.g., account creation and deletion, account privilege assignment), and use of privileges. OSs typically permits system administrators to specify which types of events should be audited and whether successful and/or failed attempts to perform certain actions should be logged.

Storing/Archiving Log Data

Due to the sensitive nature of the information that may be contained in the captured log data, it is essential that an organization take the correct steps in order to ensure that all log data adheres to today's standards of confidentiality, integrity and availability. Each organization may have a different policy regarding the information they maintain, but in every case security should be a top priority in selecting the desired method of storage and archival practices of log data. Special attention to encryption and authenticity measures should be considered in order to ensure that all data is accurate and complete. Log data that has been contaminated may generally be deemed useless, particularly when used for forensic or investigation purposes.

Monitoring Log Data

Of course log information is most useful when it is reviewed on a regular basis. Traditionally log analysis has been treated as a low priority task by both administrators and management because of the many other pressing duties in which they are responsible for. Often times it is also treated as a reactive task rather than a pro-active measure. Many administrators tasked with the duties of log analysis have received no formal training and are not provided with effective tools, which are necessary to automate and efficiently review data.

In order to make effective use of the valuable information contained within the log data, it is vital to make log analysis a priority and provide effective training and the necessary tools for the job. Log management software can quickly and efficiently parse log data and can be an invaluable tool for finding particular patterns of interest. The practice of log analysis is a vital troubleshooting and security tool and should be a key duty in any IS/IT environment. Without sound processes for log analysis, the value of the logs are significantly reduced.

Below is a "List of Compensating Controls" taken from CISA Examination Textbooks, Volume 1: Theory by S. Rao Vallabhaneni, pp 222-224.

Audit trails provide "...the ability to trace a transaction from its initiation to final disposition including all intermediate points."

Content of Transaction Logs

“tx” = transaction

1. Application tx log: tx code; record type; date and time stamp; user ID; terminal ID; tx amount; tx activity; other.
2. Database log: before and after images; I/O file information; access errors; date and time stamp; tx type and ID; terminal ID; user ID; programs used or called; access authorizations; I/O messages; list of data blocks read.
3. Operating system console log: job ID; date and time stamp; job run times (start and finish); I/O files used; programs used; media and devices used; job completion codes; operator interventions; system diagnostic messages; file backup and recovery/restore times.
4. Telecom log: originating terminal ID; transmission line ID; job ID; com port ID; date and time stamp; application type (e.g., CICS); session type (bound or not); session start and finish times; transmission error messages; user authorization code; tx ID; message ID; control totals; port/node ID; dial back phone number; messages awaiting transmission.
5. Access control security log: user ID; terminal ID; date and time stamp; tx ID; data sets used; job ID; signon ID modifications; access rule modifications; type of security violations; unauthorized access attempts and messages; invalid attempts of password; last activity date.
6. Job accounting log: job name and ID; job run time with start and finish times; date and time stamps; I/O files used; programs used; user information; job completion codes; control totals for records; I/O devices used; CPU time used; other.
7. Problem management log: problem number; problem type and description; problem report date and time; ID of reporter; problem resolution status with dates; name of original help desk person; dates and descriptions of intermediate and final solutions.
8. Change management log: change number, type and description; requestor name, ID, time and date stamp; change reason; change status code with action dates, names of persons who worked on the change and final disposition and user signoff.
9. Hardware preventive maintenance log: date and time of device failure; serial number and location of the device; time of service request; time when service arrived; time when problem was fixed; description of problem cause; nature of repair and cost; warranty period.
10. System management log: date and time stamps; datasets accessed; datasets renamed; datasets scratched; system paging activity; job CPU times; job/step termination record; data lost record; job using tape bypass label processing; remote user access.

11. Job run log: date and time stamp; job number; job source IS; job completion code; job errors; person submitting the job; job due out of time; job completion time; job status.

The **Open Systems Interconnection Basic Reference Model** (*OSI Reference Model* or *OSI Model* for short) is a layered, abstract description for communications and computer [network protocol](#) design, developed as part of the [Open Systems Interconnection](#) (OSI) initiative. It is also called the **OSI seven layer model**. The layers, described below, are, from top to bottom, Application, Presentation, Session, Transport, Network, Data Link, and Physical. A layer is a collection of related functions that provides services to the layer above it and receives service from the layer below it. For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of the path.

Even though newer [IETF](#) and [IEEE](#) protocols, and indeed OSI protocol work subsequent to the publication of the original architectural standards that have largely superseded it, the OSI model is an excellent place to begin the study of network architecture.

Contents

- [1 History](#)
- [2 Description of OSI layers](#)
 - [2.1 Layer 7: Application layer](#)
 - [2.2 Layer 6: Presentation layer](#)
 - [2.3 Layer 5: Session layer](#)
 - [2.4 Layer 4: Transport layer](#)
 - [2.5 Layer 3: Network layer](#)
 - [2.6 Layer 2: Data Link layer](#)
 - [2.6.1 WAN Protocol Architecture](#)
 - [2.6.2 IEEE 802 LAN Architecture](#)
 - [2.7 Layer 1: Physical layer](#)
- [3 Interfaces](#)
- [4 Examples](#)
- [5 See also](#)
- [6 External links](#)
- [7 References](#)

History

In 1977, work on a layered model of network architecture, which was to become the OSI model, started in the [American National Standards Institute](#) (ANSI) working group on Distributed Systems (DISY).^[1] With the DISY work and worldwide input, the [International Organization for Standardization](#) (ISO) began to develop its OSI networking suite.^[2] According to ([Bachman](#)), the term "OSI" came into use on 12 October 1979. OSI has two major components: an abstract model of networking (the Basic Reference Model, or seven-layer model) and a set of concrete

protocols. The standard documents that describe OSI are for sale and not currently available online.

Parts of OSI have influenced Internet protocol development, but none more than the abstract model itself, documented in ISO 7498 and its various addenda. In this model, a networking system is divided into layers. Within each layer, one or more entities implement its functionality. Each entity interacts directly only with the layer immediately beneath it, and provides facilities for use by the layer above it.

In particular, Internet protocols are deliberately not as rigorously architected as the OSI model, but a common version of the [TCP/IP model](#) splits it into four layers. The Internet Application Layer includes the OSI Application Layer, Presentation Layer, and most of the Session Layer. Its End-to-End Layer includes the graceful close function of the OSI Session Layer as well as the Transport Layer. Its Internetwork Layer is equivalent to the OSI Network Layer, while its Interface layer includes the OSI Data Link and Physical Layers. These comparisons are based on the original seven-layer protocol model as defined in ISO 7498, rather than refinements in such things as the Internal Organization of the Network Layer document.

Protocols enable an entity in one host to interact with a corresponding entity at the same layer in a remote host. Service definitions abstractly describe the functionality provided to an (N)-layer by an (N-1) layer, where N is one of the seven layers inside the local host.

Description of OSI layers

| OSI Model | | | |
|--------------|-----------|---------------------------------|--|
| | Data unit | Layer | Function |
| Host layers | Data | 7. Application | Network process to application |
| | | 6. Presentation | Data representation and encryption |
| | | 5. Session | Interhost communication |
| | Segments | 4. Transport | End-to-end connections and reliability (TCP) |
| Media layers | Packets | 3. Network | Path determination and logical addressing (IP) |
| | Frames | 2. Data link | Physical addressing (MAC & LLC) |
| | Bits | 1. Physical | Media, signal and binary transmission |

Layer 7: Application layer

The [application layer](#) interfaces directly to and performs common application services for the application processes; it also issues requests to the presentation layer. Note carefully that this *layer provides services to user-defined application processes, and not to the end user. For example, it defines a file transfer protocol, but the end user must go through an application process to invoke file transfer. The OSI model does not include human interfaces.* The common application services sublayer provides functional elements including the Remote Operations Service Element (comparable to Internet Remote Procedure Call), Association Control, and Transaction Processing (according to the [ACID](#) requirements).

Above the common application service sublayer are functions meaningful to user application programs, such as messaging (X.400), directory (X.500), file transfer (FTAM), virtual terminal (VTAM), and batch job manipulation (JTAM). These contrast with user applications that use the *services* of the application layer, but are not part of the application layer itself.

1. File Transfer applications using FTAM (OSI protocol) or FTP (TCP/IP Protocol)
2. Mail Transfer clients using X.400 (OSI protocol) or SMTP/POP3/IMAP (TCP/IP protocols)
3. Web browsers using HTTP (TCP/IP protocol); no true OSI protocol for web applications

Layer 6: Presentation layer

The [Presentation layer](#) doesn't alter the session layer functions as it transfers requests between the layers. It translates the data from the session layer to the application layer and provides a standard interface for the application layer. [MIME](#) encoding, **data encryption** and similar manipulation of the presentation are done at this layer to present the data as a service or protocol that the developer sees fit. Examples of this layer are converting an [EBCDIC](#)-coded text [file](#) to an [ASCII](#)-coded file, or [serializing objects](#) and other [data structures](#) into and out of [XML](#).

Layer 5: Session layer

The [Session layer](#) controls the dialogues/connections (sessions) between computers. It establishes, manages and terminates the connections between the local and remote application. It provides for [full-duplex](#), [half-duplex](#), or simplex operation, and establishes checkpointing, adjournment, termination, and restart procedures. The OSI model made this layer responsible for "graceful close" of sessions, which is a property of [TCP](#), and also for session checkpointing and recovery, which is not usually used in the Internet protocols suite. Session layers are commonly used in application environments that make use of remote procedure calls (RPCs).

[iSCSI](#), which implements the Small Computer Systems Interface ([SCSI](#)) encapsulated into TCP/IP packets, is a session layer protocol increasingly used in [Storage Area Networks](#) and internally between processors and high-performance storage devices. iSCSI uses TCP for guaranteed delivery, and carries SCSI command descriptor blocks (CDB) as payload to create a virtual SCSI bus between iSCSI initiators and iSCSI targets.

Layer 4: Transport layer

The [Transport layer](#) provides transparent transfer of [data](#) between end users, providing reliable data transfer services to the upper layers. The transport layer controls the reliability of a given link through flow control, segmentation/desegmentation, and error control. Some protocols are state and connection oriented. This means that the transport layer can keep track of the segments and retransmit those that fail.

Although it was not developed under the OSI Reference Model and does not strictly conform to the OSI definition of the Transport Service, the best known example of a layer 4 protocol is the [Transmission Control Protocol](#) (TCP). The transport layer is the layer that converts messages into TCP segments or [User Datagram Protocol](#) (UDP), [Stream Control Transmission Protocol](#) (SCTP), etc. packets.

Of the actual OSI protocols, not merely protocols developed under the model, there are five classes of [transport protocols](#), ranging from class 0 (which is also known as **TP0** and provides the least error recovery) to class 4 (which is also known as **TP4** and is designed for less reliable networks, similar to the Internet). Class 4 is closest to TCP, although TCP contains functions, such as the graceful close, which OSI assigns to the Session Layer.

Perhaps an easy way to visualize the Transport Layer is to compare it with a Post Office, which deals with the dispatch and classification of mail and parcels sent. Do remember, however, that a post office manages the outer envelope of mail. Higher layers may have the equivalent of double envelopes, such as cryptographic presentation services that can be read by the addressee only. **Roughly speaking, tunneling protocols operate at the transport layer, such as carrying non-IP protocols such as IBM's SNA or Novell's IPX over an IP network, or end-to-end encryption with IPsec.** While [Generic Routing Encapsulation](#) (GRE) might seem to be a network layer protocol, if the encapsulation of the payload takes place only at endpoint, GRE becomes closer to a transport protocol that uses IP headers but contains complete frames or packets to deliver to an endpoint. [L2TP](#) carries [PPP](#) frames inside transport packets. The habal of the layer is one of the most applicable in technology industry; it gives multiple habal layer in short habas.

Layer 3: Network layer

The [Network layer](#) provides the functional and procedural means of transferring variable length [data](#) sequences from a source to a destination via one or more networks while maintaining the [quality of service](#) requested by the Transport layer. The Network layer performs network [routing](#) functions, and might also perform fragmentation and reassembly, and report delivery errors. [Routers](#) operate at this layer—sending data throughout the extended network and making the Internet possible. This is a logical addressing scheme – values are chosen by the network engineer. The addressing scheme is hierarchical. The best known example of a layer 3 protocol is the [Internet Protocol](#) (IP). Perhaps it's easier to visualize this layer as managing the sequence of human carriers taking a letter from the sender to the local post office, trucks that carry sacks of mail to other post offices or airports, airplanes that carry airmail between major cities, trucks that distribute mail sacks in a city, and carriers that take a letter to its destinations. Think of fragmentation as splitting a large document into smaller envelopes for shipping, or, in the case of the network layer, splitting an application or transport record into packets.

Layer 2: Data Link layer

The [Data Link layer](#) provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the Physical layer. Originally, this layer was intended for point-to-point and point-to-multipoint media, characteristic of wide area media in the telephone system. Local area network architecture, which included broadcast-capable multi-access media, was developed independently of the ISO work, in [IEEE Project 802](#). IEEE work assumed sub layering and management functions not required for WAN use. In modern practice, only error detection, not flow control using sliding window, is present in modern data link protocols such as [Point-to-Point Protocol](#) (PPP), and, on local area networks, the IEEE 802.2 LLC layer is not used for most protocols on Ethernet, and, on other local area networks, its flow control and acknowledgment mechanisms are rarely used. Sliding window flow control and acknowledgment is used at the transport layers by protocols such as [TCP](#), but is still used in niches where [X.25](#) offers performance advantages.

Both WAN and LAN services arrange bits, from the physical layer, into logical sequences called frames. Not all physical layer bits necessarily go into frames, as some of these bits are purely intended for physical layer functions. For example, every fifth bit of the [FDDI](#) bit stream is not used by the data link layer.

WAN Protocol Architecture

[Connection-oriented](#) WAN data link protocols, in addition to framing, detect and may correct errors. They also are capable of controlling the rate of transmission. A WAN data link layer might implement a [sliding window](#) flow control and acknowledgment mechanism to provide reliable delivery of frames; that is the case for [SDLC](#) and [HDLC](#), and derivatives of HDLC such as [LAPB](#) and [LAPD](#).

IEEE 802 LAN Architecture

Practical, [connectionless](#) LANs began with the pre-IEEE [Ethernet](#) specification, which is the ancestor of the IEEE 802.3. This layer manages the interaction of devices with a shared medium, which is the function of a [Media Access Control](#) sub layer. Above this MAC sub layer is the media-independent [IEEE 802.2 Logical Link Control](#) (LLC) sub layer, which deals with addressing and multiplexing on multi-access media.

While IEEE 802.3 is the dominant wired LAN protocol and IEEE 802.11 the wireless LAN protocol, obsolescent MAC layers include [Token Ring](#) and [FDDI](#). The MAC sub layer detects but does not correct errors.

Layer 1: Physical layer

The [Physical layer](#) defines all the electrical and physical specifications for devices. In particular, it defines the relationship between a device and a physical medium. This includes the layout of [pins](#), [voltages](#), and [cable specifications](#). [Hubs](#), [repeaters](#), [network adapters](#), [Host Bus Adapters](#) (HBAs used in [Storage Area Networks](#)) and more.

To understand the function of the physical layer in contrast to the functions of the data link layer, think of the physical layer as concerned primarily with the interaction of a single device

with a medium, where the data link layer is concerned more with the interactions of multiple devices (i.e., at least two) with a shared medium. The physical layer will tell one device how to transmit to the medium, and another device how to receive from it (in most cases it does not tell the device how to connect to the medium). Obsolescent physical layer standards such as [RS-232](#) do use physical wires to control access to the medium.

- Establishment and termination of a connection to a communications medium.
- Participation in the process whereby the communication resources are effectively shared among multiple users. For example, contention resolution and flow control.
- Modulation, or conversion between the representation of digital data in user equipment and the corresponding signals transmitted over a communications channel. These are signals operating over the physical cabling (such as copper and optical fiber) or over a radio link.

The major functions and services performed by the physical layer are:

[Parallel SCSI](#) buses operate in this layer, although it must be remembered that the logical [SCSI](#) protocol is a transport-layer protocol that runs over this bus. Various physical-layer Ethernet standards are also in this layer; Ethernet incorporates both this layer and the data-link layer. The same applies to other local-area networks, such as [Token ring](#), [FDDI](#), and [IEEE 802.11](#), as well as personal area networks such as [Bluetooth](#) and [IEEE 802.15.4](#).

Interfaces

In addition to standards for individual protocols in transmission, there are also interface standards for different layers to talk to the ones above or below (usually operating-system-specific). For example, [Microsoft Windows' Winsock](#), and [Unix's Berkeley sockets](#) and [System V Transport Layer Interface](#), are interfaces between applications (layers 5 and above) and the transport (layer 4). [NDIS](#) and [ODI](#) are interfaces between the media (layer 2) and the network protocol (layer 3).

OSI Service Specifications are abstractions of functionality commonly present in programming interfaces.