**Chapter 1: An Introduction to DataPower SOA Appliances**

**Overview**

*Let’s get one thing straight right from the top—these are not your mother’s appliances!*

Let’s use that opening statement as a springboard for our discussion on exactly what SOA appliances are, how they are used, and how they are similar and dissimilar to traditional household appliances. The use of the term *appliance* to describe this new class of IT products is no accident. It is meant to convey certain parallels to the term that is familiar to us. Think about it—what are the characteristics of your typical household appliances? Visualize the appliances of yesteryear rather than the more complex ones we see on the market today. Certain attributes should come to mind:

* **Purpose-built—** Appliances at home are typically for specialized uses—one for washing clothes, one for keeping food cold, and so on.
* **Simple—** Most appliances have few knobs and controls. They have simple designs due to the dedicated purpose for which they are designed. They are also reliable, so they don’t need to be serviced or replaced often.

Get the picture? Now let’s move the discussion to a realm where we as IT professionals are more comfortable—for many, that is *not* the realm of domestic chores!

There is a current trend in IT shops to use specialized appliances wherever possible. This is due to several factors, the primary ones being total cost of ownership (TCO), return on investment (ROI), performance, integration, ease of use, and security. To get started, we introduce you to IBM’s WebSphere DataPower SOA appliances, and then talk about how appliances can help in each of these areas. Of course, we go into much greater detail throughout this book.

## Meet the Family!

The primary[[1](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=472904439&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn01)] three products in the DataPower family are the DataPower XA35, XS40, and XI50, as shown in Figure 1-1. As you can see, the products are outwardly similar in appearance. Each is a hardened 1U rack-mount device in a tamper-proof case with four RJ-45 Ethernet ports, a DB-9 serial port, and a power switch. We are speaking about the base configuration—there are options available, such as adding a Hardware Security Module, which could alter the outward physical profile. There are also replaceable fan trays, batteries, power supplies, and compact flash cards or hard drives.

[](javascript:PopImage('IMG_2','http://images.books24x7.com/bookimages/id_30903/01fig01_alt.jpg','1258','687'))

Figure 1-1: The DataPower product family.

In the following sections, we discuss the feature set for each model and then move on to scenarios in which appliances can be of great value before taking a closer look at what’s under the covers.

### DataPower XA35

The DataPower XA35 (on the bottom in Figure 1-1) is the entry level product in the line and most representative of the beginnings of the product and DataPower company. The appliance is green, which represents its primary function: to make XML “go faster.” This is also the impetus behind the designation of the “A” in XA; it stands for acceleration. The XA35 is at its core a highly efficient XML processing engine. It makes use of DataPower’s purpose-built features, such as optimized caches and dedicated SSL hardware to process XML at near wire-speed.

The XA35 is a hardened appliance, but it has limited security processing functionality; for example, it does not have the full XML threat protection or encryptiondigital signature capabilities as the other models that we discuss. For these reasons, it generally sits behind the DMZ,[[2](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=472904439&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn02)] in the trusted zone to augment the processing of XML files. For example, it may be configured to do validation and transformation of XML before it reaches (or for traffic flowing between) the backend servers. It should be used in-line in the network topology, not as a co-processor hanging off a particular server (although this latter usage is how the appliances were first designed). A popular usage is to receive XML responses from backend servers and transform those into HTML before continuing the response to the client. It has full SSL and SNMP capabilities to fit into the network infrastructure.

### DataPower XS40

The DataPower XS40 (in the middle in Figure 1-1) is called the security appliance, and justifiably it is yellow, which represents caution or yield. The “S” in XS stands for security. This model is often found in the DMZ, as its security capabilities are extensive.

The XS40 has all the capabilities of the XA35, plus the following:

* Encryptiondecryption utilizing purpose-built hardware for RSA operations
* Digital signature creationverification
* Fine grained Authentication, Authorization, and Auditing (AAA)
* Full XML threat protection
* Tivoli® Access Manager (TAM) integration option
* Hardware Storage Module (HSM) option
* Dynamic routing
* Message filtering
* Fetching content from remote servers
* MIME, DIME, and Message Transmission Optimization Mechanism (MTOM) attachment processing
* XML Generation 4 (XG4) accelerator module option
* Web services management
* Service level monitoring

### DataPower XI50

The DataPower XI50 (at the top in Figure 1-1) is truly the star of the show. It is the integration appliance, as represented by the “I” in XI, and it is IBM blue (what else!) in color. Due to its integration capabilities, it is often found in the backend private network, functioning in an ESB capacity but is just as suitable for the DMZ. The majority of this book focuses on the XI50, as it is a superset of the other two models.

The XI50 has all the features of the XS40 (and hence the XA35) plus the following:

* WebSphere MQ client option
* WebSphere Java Message Service (JMS) Jetstream protocol connectivity
* TIBCO Enterprise Message Service (EMS) connectivity
* IBM IMS Connect client
* Database option (DB2, Sybase, Oracle, SQL Server)
* Optimized run-time engine for non-XML transformations

This might seem like a short list compared to all the capabilities that the XS40 heaps on what the XA35 had, but these are some big-ticket items! Throughout this book, you will see just how important these features are and how to leverage them.

Now that we’ve had our brief introduction, let’s talk about where appliances are being used in corporate information technology shops, and what kinds of problems they can help solve.

[[1](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=472904439&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn01)]There are also specialized, derivative appliances, such as the XB60 Business-to-Business and XM70 Low Latency Messaging devices, which are discussed in [Appendix C](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=5390#5390), [“DataPower Evolution.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=5390#5390)

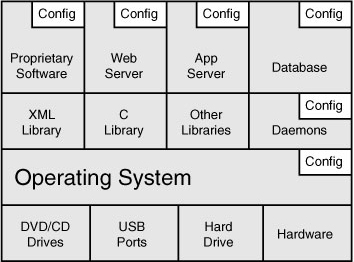
[[2](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=472904439&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn02)]A DMZ is generally the front-facing “perimeter” of a network, where client traffic enters. Because it’s the first point of entry into your network, and hackers have access, it must be hardened.

## Typical Usages of Appliances

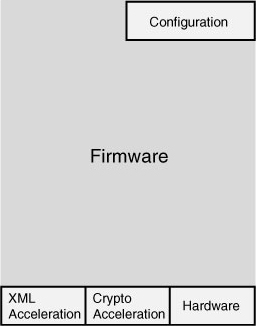
While the appliances are quite versatile and can thus be used to solve many different types of problems (and implementers have been quite creative in this regard), we find there are a few common use cases that are typical. These generally focus around security, performance, cost savings, and integration. In the following sections, we discuss each of these in more detail.

### Solving Security Problems

Let’s think about what it would take to deploy a software-based proxy product in the DMZ. Each of the layers of the ‘typical server’ shown in Figure 1-2 requires specialized skills to install and maintain. Particularly for DMZ deployments, the server hardware itself must be hardened. In highly secure environments, this can involve removing any components that might allow information to be taken from the server, such as USB ports and writeable CD/DVD drives. The operating system must also be hardened, removing components such as telnet and sendmail.[[3](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=648633428&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn03)] Often, this results in other layers of the software stack not installing or operating properly! If you are successful in installing the application software stack, it must be hardened as well. These are common requirements for high security environments such as financial companies, intelligence services, and military applications.

  
Figure 1-2: Typical server components.

Although software-based DMZ components can be hardened successfully, it is a lot of work. Compare this with the simplicity of installing a dedicated, highly secure hardware appliance, purpose built to do a few things well with fairly simple administrative interfaces, as shown in Figure 1-3.

  
Figure 1-3: High-level SOA appliance components.

The appliances are hardened out of the box. For example:

* They are designed with security in mind from the ground up, before anything else.
* They are shipped secure by default; virtually every feature is disabled, including the network adapters and administrative interfaces (except for the serial port used to do initial bootstrap). If you want something, *you* must turn it on!
* They have an encrypted file system.
* They have no Java, print services, or shareable file system.
* They are tamper-proof—backing out the screws on the case disables the appliance.
* They have specialized secure handling of crypto keys and certificates.
* They have an embedded operating system, not prone to known exposures of common OSs.
* They reject messages by default, unless specifically accepted by configured policies.

The age-old rule for the DMZ is to terminate client connections there and then proxy connections to the backend from the trusted DMZ servers. However, in the field we find even more stringent security policies that do not warrant *any* traffic (even proxied through these secure intermediaries) to the backend until the client is authenticated and authorized. This is referred to as [*perimeter security*](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1856#1856) and is an increasingly common requirement, driving sales of DMZ security products such as TAM. Later, we show how DataPower appliances can also solve this problem.

Another requirement for DMZ components is to virtualize or hide the implementation details of backend servers and applications. Typical DMZ products interact only with the protocol layer of the network stack, so they can hide things like hostname/IP, ports, and URIs, whereas XML-centric application proxies such as DataPower appliances can virtualize on a much more intelligent basis and can analyze the entire message stream.

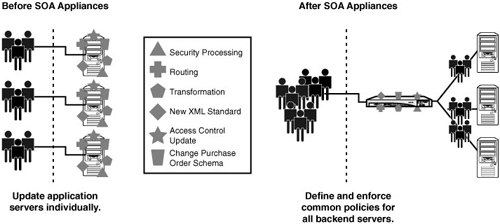
A strong reason for using these types of appliances is the burgeoning risk of systems becoming compromised by XML-based threats. Just as once upon a time we felt HTTP to be innocuous, today we are susceptible to underestimating what can be done by virtue of XML. In [Chapter 20](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3695#3695) [“XML Threats,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3695#3695) we show how entire infrastructures can be brought down using small, simple, well-formed XML files. Only hardware appliances have the processing power to check for the many variations of XML threats.

Another common security problem is a mismatch in the specification levels or credential formats of various technologies across large corporate IT infrastructures. For example, consider a marketing IT silo running on Microsoft®.NET using WS-Security 1.0 and SPNEGO credentials for identity and a manufacturing silo using IBM WebSphere Application Server (WAS), WS-Security 1.1, and LTPA credentials for identity. In today’s ESB-driven SOA architectures, a single transaction may have to pass through both environments, so this presents challenges. Because DataPower appliances incorporate a wide range of the latest specification implementations and credential formats, they can be used to transform messages and credentials to fit the target each step of the way. Notice that this can be used to achieve cross-platform single-signon (SSO), although that also depends on other factors such as having a common registry.

### To Lower Total Cost of Ownership (TCO)

Refer back to the scenario in Figure 1-2, where there are numerous skills required to install and maintain a typical server and software stack. Now think of this in terms of the staff required and cost to the organization. With self-contained appliances where the operating system and file system characteristics are irrelevant from an administrative perspective, this becomes much less work. The function of the appliances is dedicated and streamlined, hence the administrative tasks and interfaces tend to be as well. For example, in the scenario in Figure 1-2, you have to continually install fixes and updates at every layer of the stack. However, for appliances, you typically do this by uploading a small firmware update and rebooting, which takes only minutes. In the server scenario, you have multiple different administrative consoles to manage the layers of the stack; with the appliances, you have only one console.

The TCO return does not solely manifest itself in the setup and administration of the platform. Consider the silo example in the prior section—where various areas of a corporate IT infrastructure are running Web services across different platforms, such as those from IBM, Microsoft, and BEA. If the corporation has one set of policies for security and SLM that need to be implemented across all these platforms, then it must be done multiple times, by multiple people, with skills on each platform. Not only is the configuration redundant and therefore expensive, but this problem is repeated each time it needs to change, and there is always the risk that the policy will not be implemented exactly the same on each platform, which can lead to security holes or application failures. This is depicted in Figure 1-4.

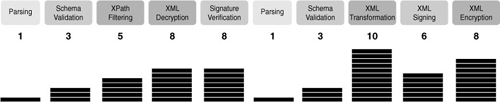
[](javascript:PopImage('IMG_5','http://images.books24x7.com/bookimages/id_30903/01fig04_alt.jpg','713','320'))  
Figure 1-4: Redundant administration versus simplified appliance model.

A more concrete example can be implemented by creating a single service that acts as a Web service proxy on the DataPower appliance, importing the WSDL files for the Web services providers on each of those backend platforms, and then applying the security and SLM policies on the proxy, thereby gaining policy definition and enforcement one time for all platforms. All this is based on standards that we discuss later, not only Web services itself, but also the accompanying standards such as WS-Security for security, WS-Policy for policy definition, WS-Addressing for endpoint resolution, and WS-Management and WSDM[[4](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=648633428&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn04)] for management.

### Enhancing Performance

XML is the foundation on which many modern architectures are built—it has evolved into SOAP for Web services and is found across the breadth and depth of the SOA stack and related specifications. Over time, it has evolved from a simple markup language to something quite complex and sophisticated. Of course, the problem as far as performance is concerned is that XML is fairly easy for humans to read, but not for computers. It is a verbose representation of data and typically requires significant resources in terms of CPU power and memory to process. This overhead is typically found in parsing the XML document into an in-memory representation and in validating the XML against its schema file.[[5](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=648633428&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn05)]

Consider the impact of parsing and validating the storm of XML/SOAP documents that hit your systems during peak production levels. Now consider the overhead of security that may be embedded in those messages—validating client identities against LDAP servers, verifying digital signatures, and decrypting encrypted data. This requires a tremendous amount of processing power and time and robs precious cycles away from what your backend systems should really be doing—focusing on transactional business logic! Also consider the absolute waste of expending these cycles for messages that come in badly formed, with schema violations or illegitimate security issues. The cycles expended on processing them and handling the errors are wasted. Figure 1-5 shows a graph demonstrating the CPU overhead of various common tasks. (Notice the parsing level is low here—the main hit when parsing is memory utilization.) Notice the impact of security operations. This can be helped somewhat with hardware-assisted acceleration, but the cost-benefit of hardware acceleration boards is often debated. Also note that abusing these security features to consume CPU resources is one way of mounting attacks.

[](javascript:PopImage('IMG_6','http://images.books24x7.com/bookimages/id_30903/01fig05_alt.jpg','879','180'))  
Figure 1-5: Security impact of common tasks.

A grand solution for this, of course, is to use appliances to do all that heavy lifting at near wire speed. As you will see when we discuss the appliance characteristics, they are amazingly fast and can handle these tasks at orders of magnitude faster than software-based solutions running on standard servers. Now focus on another scenario—one where the appliance makes sure that only clean traffic gets to the backend systems. Imagine the huge differential in available processing power on the backend if the validation and security tasks are done by the time the traffic gets there. The appliances can validate schemas, verify signatures, decrypt the data, and more. This can often result in huge performance returns, depending on considerations such as message sizes, cipher strengths, network latency, and so forth.

Speaking of message sizes, this is often another major stumbling block for Java-based software systems processing XML. In modern day real-world systems, we are now seeing huge SOAP messages on the order of hundreds of megabytes or even gigabytes in size. The conundrum is how to process these, given constraints on maximum JVM heap sizes in many platforms. Due to aggressive built-in streaming and compression, appliances can handle messages larger than their actual memory space.

On another message-related topic, consider applications that do XML transformation between differing schemas; for example, an application that consumes XML purchase orders and must understand a variety of incoming purchase order formats from business partners, and then transforms each into the one “golden” purchase order schema that this company uses. These transformations can be quite expensive to process (see Figure 1-5) and result in bloated application code. We all know that line-for-line, application code is expensive in terms of programmer time, testing, and debugging. Now consider the effect on the application if the transformations were moved out to the appliance on the frontend so that the backend application now gets only the one “golden” schema format. Yes, our application has gone on quite a diet, is less expensive to maintain, and is much faster. One field scenario consisted of a frontend cluster of Java EE applications to do such transformations to keep the cluster of business logic applications behind it lightweight. However, since this was running on a platform that charged for CPU time, and given the overhead of XML transformations shown in Figure 1-5, it was expensive. The solution was to move the transformation layer out to DataPower appliances. The result was a huge cost savings and orders of magnitude faster processing.

### Integrating Platforms

In the previous section, we discussed a scenario in which the appliance could be used to bridge differences in standards specifications (WS-Security v1.0 versus. v1.1) and identity credentials (SPNEGO versus LTPA) across systems. This is one good example of easily integrating disparate platforms, particularly when the standards and specifications are in flux. It is difficult for software-based solutions running on standard servers and products to keep up with this. On the appliance, you load a firmware update to get the latest and greatest.

### Note: Firmware Versions Used for This Book

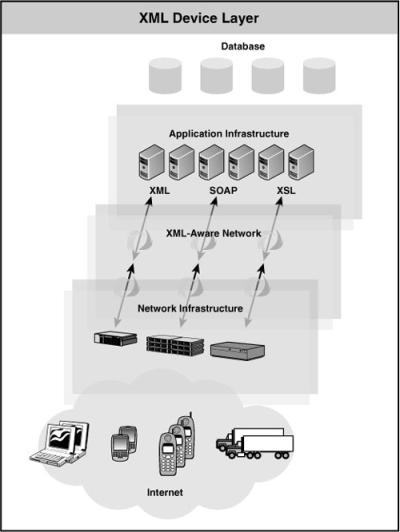
The recommendations, advice, and practices shown in this book are generally applicable to firmware versions 3.6.0 through 3.7.2 (and likely future releases) and based on the DataPower XI50. However, much of the information in this book is “timeless” in that it represents information that is generally accepted as “best practices” in our experience for most situations, and unrelated to specific firmware versions.

However, there are other issues that arise when integrating different platforms. Consider a scenario in which a medium-sized business XYZ Corp has its infrastructure running on legacy platforms and technologies, perhaps mainframe-based EDI. The business partners that they depend on have long since moved their platforms to Web services and are telling poor XYZ Corp that they can no longer afford to support XYZ’s legacy interface to that system, and they must provide a modern SOA or Web services interface or lose the business. This puts XYZ in a bad position; what will it cost to retrain its programmers, rewrite its COBOL applications, and revamp the backends to its Java EE platforms? Likely, it would be a staggering amount! A common solution to this problem is to place appliances at the front of the network as proxies, cook up a WSDL file to describe some Web services, begin receiving the ASCII SOAP messages from the now-happy business partners, and convert them on-the-fly to EBCDIC EDI or COBOL Copybook messages and send them over MQ or IMS Connect to the legacy backend. The backend does not have to change, and no programs have to be rewritten—a win-win!

Due to the variety of protocols (HTTP(S), FTP, MQ, JMS/JFAP, IMS, NFS, TIBCO, MQ, ODBC, SNMP, and so on) supported by the DataPower appliances, there is a wealth of opportunity for protocol bridging, content enrichment, and integration between platforms. Notice that the previous scenario involved message transformation. The XI50 DataPower appliance can handle either XML-to-XML or non-XML transformation scenarios, meaning that messages can be transformed to the appropriate format for any intended backend.

Another common and age-old scenario related to integrating platforms is dynamic routing. Because it is often a requirement to make dynamic routing decisions “on the edge of the network,” we have DMZ Web servers, proxies, and load balancers handle this. The problem is that they can understand only the protocol and not the payload of the message. To accomplish the goal, applications place some value in the protocol header to facilitate the content-based routing. As an example, if we want any purchase orders over one million dollars to be routed to highpriority servers, the sending application would place a cookie or attribute in an HTTP header or URL parameter. The Web server, proxy. or load balancer in the DMZ would be configured to check for this and then route the traffic accordingly. The problem with this scenario is that you have to put this hack in the applications and the HTTP payload, potentially disclose message data to attackers, and involve the sender/client. This solution doesn’t scale because if you continually do this, the HTTP header and application code bloat.

Because SOA appliances are XML-savvy and can use technologies such as XPath, they can check *inside* the message payload to look for the actual <po\_value> element rather than alter the application and HTTP header. If the message is encrypted, you don’t need to expose this by externalizing the data; you can just decrypt the message and check the value, and then route accordingly. The client in this case does not have to be complicit—the routing is truly dynamic and transparent. The XML Aware Network layer is shown in Figure 1-6.

[](javascript:PopImage('IMG_7','http://images.books24x7.com/bookimages/id_30903/01fig06.jpg','500','666'))  
Figure 1-6: XML Aware Network layer.

One last important feature in regard to the integration story is the use of appliances as ESBs. The appliances fulfill the model of an ESB by virtue of their strong routing, transformation, mediation, and protocol-switching capabilities. IBM has other ESB products capable of implementing the ESB pattern—WebSphere Message Broker (WMB) and WebSphere Enterprise Service Bus (WESB). Each of these have unique capabilities that may suit them for particular usages. Although DataPower may be thought of as a highly secure and performant ESB, the others have features that DataPower does not have in the arenas of transactionality, persistent message handling, and the capability to work in other programming languages. We discuss ESBs in [Chapter 5](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=635#635), [“Common DataPower Deployment Patterns,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=635#635) and [Chapter 9](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1276#1276), [“Multi-Protocol Gateway.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1276#1276)

[[3](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=648633428&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn03)]The Center for Internet Security (<http://cisecurity.org/>) has papers showing how to harden various platforms, as well as scoring tools to see how well your platform is hardened.

[[4](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=648633428&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn04)]WSDM (Web Services Distributed Management) is a Web service standard for managing and monitoring the status of Web services.

[[5](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=648633428&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn05)]An XML schema definition file (XSD) is a set of rules for how the file should look and what it should contain, including optional and required elements.

## A Closer Look at the DataPower Products

Now that you have a general idea what “SOA Appliances” are, and have some familiarity with the IBM offerings in this space and what they are used for, we will describe them in more detail.

### Physical Characteristics of Appliances

As stated earlier, and demonstrated in [Figure 1-1](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=32#32) appliances are “pizza-box,” rack-mountable 1U (1.75-inch thick) hardware devices. The only external interfaces are a power switch, 9-pin serial port, and four RJ-45 Ethernet ports. (Appliances with HSM will have a Pin Entry Device [PED] connector.)

### Software Architecture of Appliances

As [Figure 1-3](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=54#54) illustrates, the software architecture is simple from the user perspective. There is a customized, hardened, native-code operating system kernel that implements the appliance’s core functionality. The OS resides in firmware that can be updated by applying small firmware update files.

On top of this is a layer of functionality that is implemented in XSLT stylesheets, which are read-only and used by the system to implement certain functionality. We get into more detail in [Chapter 2](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=145#145), [“DataPower Quick Tour and Setup.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=145#145)

The next layer up the software stack consists of configurations developed by the user; these are the application proxies and processing policies to process message traffic for your applications. Configuration files and application artifacts can reside in the directory structure on the file system or they can be hosted on remote servers and retrieved and cached at start-up time so that they do not ever reside on the appliance file system (a requirement in some highly secure environments).

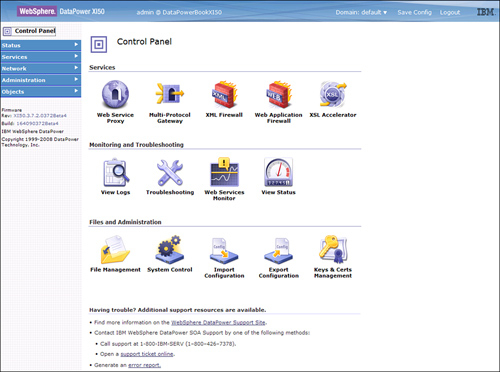
Although the operating system itself and many of the appliances’ implementation details are custom and proprietary, outwardly, the appliances are built on a standards-based model. A few important ones are listed here. These are based on a single foundation, XML.

* **XML—** A general purpose specification for creating other markup languages—and many are built upon it, such as MathML (a markup language to describe mathematics). It is a combination of data and metadata, consisting of tagged elements to not only show the data but to describe and delineate it; for example, <po\_number>12345</po\_number>.
* **XSD—** A set of rules that an XML file must conform to. So if you want to define a purchase order XML file to use with your applications, you can create an XSD file to be used to validate those incoming purchase order XML files to ensure they have the proper structures.
* **SOAP—** A message format used by Web services for sending and receiving XML-based messages. It is more sophisticated than “normal” XML in that its construct provides for a message header and body, among other things.
* **WSDL—** A language for describing Web services. It defines the services, ports, bindings, and operations that constitute the Web service, along with the endpoint information (hosts, ports, URIs) and perhaps other metadata such as policy information.
* **XPath—** XPath for XML is somewhat analogous to SQL for databases.[[6](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn06)] XPath allows for searching and retrieving information (nodesets) from XML documents based on some criteria.
* **XSLT—** An XML language for transforming XML documents from one format to another. If you want to transform a vendor’s XML purchase order format to your own company’s XML format, you can write a set of instructions in XSLT to do so.
* **EXSLT—** A community extension to XSLT to provide a library for things like string manipulation, date/time functions, and other miscellaneous library functions.

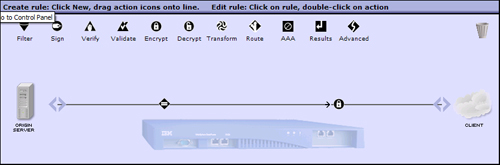
### Administrative Model

As part of the “secure by default” DataPower mantra, all remote administrative interfaces are shut down by default. The only way to enable them is by bootstrapping the appliance via the serial port. We show how to do this in [Chapter 2](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=145#145). After you do this, you have several options for administrative interfaces. These are described in detail in [Chapter 12](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2003#2003) [“Device Administration”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2003#2003) and [Chapter 13](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2191#2191) [“Alternate Management Interfaces,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2191#2191) but we give a brief overview in the following list:

* **Command-shell Admin—** This can be accessed using telnet, secure-shell (SSH), or the serial port. The Command Line Interface (CLI) is an IOS-like interface, which will be familiar to many network administrators. In the most ultra-secure environments, all remote administrative interfaces are disabled, forcing all administration to be done only by those with physical access to the appliances in the datacenter. For security purposes, telnet normally remains disabled.
* **XML Management Interface—** The XML Management interface provides a way to administer the appliance and obtain status information via XML-based SOAP requests. There are several different specifications that can be used, including DataPower’s own SOAP Configuration Management,[[7](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn07)] WS-Management, and WSDM. This interface is commonly used for automated, programmatic, or custom approaches to administration.
* **WebGUI Admin Console—** This is a standard browser-based administrative interface. It is the most commonly used way to administer the appliances. However, in some high security or production environments, browser-based administration is not permitted and is allowed only in development environments as a convenience for developers. You can see in Figure 1-7 that the WebGUI is well laid out, attractive, and intuitive.[[8](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn08)]

[](javascript:PopImage('IMG_8','http://images.books24x7.com/bookimages/id_30903/01fig07_alt.jpg','995','740'))  
Figure 1-7: DataPower Web Admin console.

The administrative WebGUI is not only used for administering the appliance, it is also used to create the application proxies that are the raison d’être (justification for existence) for the product. You can use the drag-and-drop capabilities of the Processing Policy editor to create work-flow type rules for requests and responses, to carry out various actions as traffic flows through the device. Figure 1-8 shows the simplicity of dragging an Encrypt Action from the upper palette row of actions to the processing rule to encrypt a message as it passes through to its destination. From here, only the certificate to be used for the encryption needs to be identified, although there are many other advanced options that can be chosen, such as the encryption algorithm to use. Compare the ease of this to creating policies to encrypt a message on other platforms (and then factor in the performance difference). Notice in this figure that the other types of actions can be just as easily applied for tasks such as message filtering, creating or validation digital signatures, transforming messages, dynamic routing, and AAA. The Advanced Action contains a great deal more.

[](javascript:PopImage('IMG_9','http://images.books24x7.com/bookimages/id_30903/01fig08_alt.jpg','835','276'))  
Figure 1-8: Drag-and-drop policy editor.

Often, the browser-based console is used only in development environments for easily building proxies, and from there, automated, scripted processes are used to deploy these configurations to test, QA and production environments, leveraging either the command-line, or SOAP-based administrative interfaces. These techniques are described in [Chapter 15](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2720#2720), [“Build and Deploy Techniques.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2720#2720)

### Programming Model

### As shown in the previous section, most of the work in configuring the appliances is done using the friendly drag-and-drop paradigm of the Processing Policy editor. For any customized scenarios not covered by the GUI, the devices can be programmed.

As the appliance is XML-centric, the custom programming model for DataPower is XSLT, which is a full Turing-complete programming language. Any custom programming is done in this language.

XPath is an important technology for these XML-centric products. Aside from custom programming done in XSLT, XPath expressions are used frequently in building configurations using the WebGUI. For example, if you are building a policy to sign and/or encrypt selected nodesets in an XML or SOAP document, you simply provide DataPower an XPath expression so that it can locate those nodesets. For nonprogrammer types, the DataPower WebGUI provides an easy-to-use XPath tool that enables you to load a sample document and click on the element or nodeset, and the XPath statement is generated for you.

The DataPower appliances offer much more than what standard XSLT and EXSLT have in their libraries. The appliances support crypto operations and many different protocols that are outside the domain of XSLT and EXSLT. To provide for custom programming that leverages the full scope of functionality on the appliances, they include a complete library of extension functions and elements that can be used for XSLT custom programming. These are covered in the chapters in [Part VI](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4077#4077), “[DataPower Development](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4077" \l "4077" \t "_parent).”

Of course, all the power of XML, SOAP, and many of the WS-\* specifications/standards are available on the appliance. Some of the key WS- specifications are

* **WS-Security—** A specification to enable message integrity, message privacy, and non-repudiation,[[9](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch01fn09)] typically using digital signatures and encryption.
* **WS-Addressing—** A specification to enable Web services to communicate endpoint and addressing information between themselves.
* **WS-Policy—** A specification that allows Web services to advertise and enforce policies for things like security and quality of service.
* **WS-ReliableMessaging—** A specification that enables Web services to reliably transmit SOAP messages, even when there are problems in the infrastructure that would otherwise lead to failure.

[[6](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn06)]A newer and related XML specification named XQuery is much closer to true SQL capability.

[[7](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn07)]This has in the past been referred to as SOMA, which is shorthand for SOap MAnagement.

[[8](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn08)]This is the admin interface for an XI50; the other models will have fewer features.

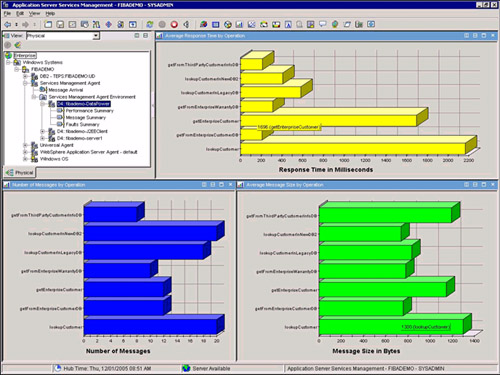
[[9](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=981743249&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch01fn09)]Many security experts find non-repudiation to be a weak concept.

**DataPower as a Member of the Network Infrastructure**

At their physical core, the DataPower appliances are network devices. Certainly, by looking at them, this would be one’s presumption. In Figure 1.1, the most apparent feature is the set of four network interface jacks on the front of the appliance. On the appliance, these are labeled MGMT, ETH0, ETH1, and ETH2. They can be split up any way you choose; for example, it is common to dedicate the Management port to the administrative subnet. From there, the remaining three can be split up so that two receive client traffic and the third connects to the backend private network, thereby segregating the network data for network security.

There are also a number of network protocols supported on the appliance. These include HTTP, HTTPS, FTP, FTPS, SFTP, NFS, MQ, MQ/SSL, JMS, and Tibco EMS for application traffic, and SNMP, SMTP, sFTP, and others for administrative usage.

We’ve mentioned SNMP a few times, which is ubiquitous and useful for infrastructure monitoring. The appliance comes with SNMP MIB files that can be imported into your monitoring tools to set up monitoring policies, and the appliances can send out SNMP traps when critical events occur. Monitoring can also be achieved by using SOAP, as is the case with the integration with Tivoli ITCAM for SOA (see Figure 1-9). There are also objects built in that are useful for monitoring and auditing, such as message count and duration monitors and sophisticated service-level management tools. Most logging is done off-device, utilizing protocols such as syslog and syslog-NG, or by writing logs to a remote NFS mount. (DataPower never shares its own file system, but can connect to shared file systems on other servers.) There is a full suite of logging formats and protocols for your use, as well as a model for specifying event notifications on various levels of granularity.

[](javascript:PopImage('IMG_10','http://images.books24x7.com/bookimages/id_30903/01fig09_alt.jpg','688','516'))  
Add a note hereFigure 1-9: Monitoring DataPower appliances with Tivoli ITCAM for SOA.

Add a note hereIncluded with the appliances is a utility for managing multiple devices, ITCAM SE for DataPower, which is based on a cut-down version of the Tivoli ITCAM product built on the Tivoli EnterpriseTM Monitoring Server (TEMS). This fat-client utility is installed on a server or workstation and enables appliances to be grouped into managed sets in order to keep their firmware levels and configurations in sync. This can be used to cluster application proxies for high availability and better levels of service. It also backs up the configurations when it detects changes.

Similar management features are also included in the WAS 7.0 administrative console. Both utilities are covered in [Chapter 29](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=5160#5160), [“Multiple Device Management Tools.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=5160#5160)

## Summary

Add a note hereThis chapter served as an introduction and overview of the IBM WebSphere DataPower SOA appliances. We introduced you to the product family and ran through some use cases where the strengths of this platform are emphasized, and then took a closer look and discussed at a glance how the appliances fit in with the rest of the network infrastructure. We expand on all these principles in the following chapters. Although we cannot cover every aspect of these unique devices, we hope to describe those most-often used in your enterprise deployments.