IBM WebSphere DataPower SOA Appliance Handbook

Bill Hines, John Rasmussen, Jaime Ryan,
Simon Kapadia, Jim Brennan

Forewords by Dr. Eugene Kuznetsov,
Jerry Cuomo, and Kyle Brown

IBM Press Program Managers: Steven M. Stansel, Ellice Uffer

Cover design: IBM Corporation

Associate Publisher: Greg Wiegand

Marketing Manager: Kourtnaye Sturgeon

Publicist: Heather Fox

Acquisitions Editor: Katherine Bull

Development Editor: Ginny Bess Munroe

Managing Editor: Kristy Hart

Designer: Alan Clements

Senior Project Editor: Lori Lyons

Copy Editor: Editorial Advantage

Indexer: WordWise Publishing Services LLC

Compositor: TnT Design

Proofreader: San Dee Phillips

Manufacturing Buyer: Dan Uhrig

Published by Pearson plc

Publishing as IBM Press

IBM Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U. S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com.

For sales outside the U. S., please contact:

International Sales
international@pearsoned.com.

Library of Congress Cataloging-in-Publication Data

IBM websphere datapower SOA appliance handbook / Bill Hines ... [et al.].
p. cm.
ISBN 978-0-13-714819-6
1. WebSphere. 2. Web site development. I. Hines, Bill.
TK5105.8885.W43I265 2008
004.6—dc22

2008042957

ISBN-13: 978-0-13-714819-6

ISBN-10: 0-13-714819-4

Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.
First printing December 2008

Foreword by Eugene Kuznetsov

> "The proper planning of any job is the first requirement. With limited knowledge of a trade, the job of planning is doubly hard, but there are certain steps that any person can take towards proper planning if he only will."
> —Robert Oakes Jordan, *Masonry*

I founded a company called DataPower® in the spring of 1999 to build products based on several distinct ideas. The first idea involved applying reconfigurable computing and dynamic code generation to the problem of integrating disparate applications. The second idea centered on the concept of data-oriented programming (DOP) as the means to achieve direct and robust data interchange. The third idea involved delivering middleware as a network function, enabled by the DOP technology and inspired by the successful models of ubiquitous connectivity. The product's journey since has been remarkable, and this great book is another milestone for the entire team behind DataPower. Before more discussion of the book itself, a few words on these three ideas.

Rapidly adapting to change is key for everything and everyone in today's world, and IBM SOA appliances are no exception. Whether it's a policy, a transformation map, a schema, or a security rule, DataPower will try to put it into effect with as little delay and interruption as possible. Popular methods for maintaining this kind of flexibility come with a large performance penalty. However, by dynamically generating code and reconfiguring hardware based on the current message flow, it became possible to achieve both flexibility and near-optimal performance. At any given point, the device operates as a custom engine for a particular task, but when the task changes, it can rapidly become a different custom engine underneath the covers.

This dynamic adaptability is especially useful when combined with DOP. Stated briefly, DOP emphasizes formally documenting data formats and using them directly, instead of encapsulation or abstraction, to integrate or secure different modules or systems. Today, XML is probably one of the most successful and readily recognized examples of DOP, but the principles are more universal than any particular technology. Another example of DOP is the way DataPower XI50 processes binary data, by using high-level format descriptors instead of adaptors.

These, in turn, enable the creation of network hardware (also known as appliance) products that operate on whole application messages (rather than network packets) to integrate, secure, or control applications. Greater simplicity, performance, security, and cost-effectiveness were envisioned—and are now proven—with the appliance approach. Beyond the appliance design discipline, the success of IP & Ethernet networking in achieving universal connectivity has much to teach about the best way to achieve radically simplified and near-universal application integration.

Reading this book will enable you to benefit from the previous three ideas in their concrete form: the award-winning IBM products they became. From basic setup to the most powerful advanced features, it covers DataPower appliances in a readable tone with a solid balance of theory and examples. For example, Chapter 6 does a great job in explaining the big-picture view of device operation, and Chapter 22 gives a detailed how-to on extending its capabilities. With some of the most experienced hands-on DataPower practitioners among its authors, it provides the kind of real-world advice that is essential to learning any craft.

When learning IBM DataPower, there is one thing that may be more helpful and rewarding than remembering every particular detail, and that is developing an internal "mental model" of how the devices are meant to operate and fit into the environment. Especially when troubleshooting or learning new features, this "mental model" can make device behavior intuitive. Reading the following pages with an eye toward not just the details but also this mental model will speed both productivity and enjoyment.

In conclusion, I would like to use this occasion to thank the entire team, past and present, who made and continues to make DataPower possible. Their work and the passion of DataPower users is an inspiring example of how great people and a powerful idea can change the world for the better.

—Eugene Kuznetsov, Cambridge, MA Founder of DataPower Technology, Inc. served as President, Chairman, and CTO at various points in the company's history, and then served as director of Product Management and Marketing, SOA Appliances at IBM Corporation.

Figure 1 DataPower's first office is on the right. (*Photo courtesy of Merryman Design.*)

Foreword by Jerry Cuomo

It all started when I was asked to co-host an IBM Academy Conference on "Accelerators and Off-Loading" in 2004. I was feeling a little out of my element, so I decided to take some of the focus off me and put it on others. I had been reading about some of the new XML-centered hardware devices and was intrigued. I have always been interested in system performance. With XML dominating our emerging workloads (e.g., Service Oriented Architecture), the impact of XML performance on system performance was becoming increasingly important. Hence, I thought it would be a good idea to invite a handful of these XML vendors to our conference.

At the conference, the DataPower presentation was quite different from the others. It wasn't about ASICs or transistors; it was about improving time to value and total cost of operation. The DataPower presentation focused on topics that were also near and dear to me, such as systems integration, configuration over programming, and the merits of built-for-purpose systems. In essence, Eugene Kuznetsov, the DataPower founder and presenter, was talking about the value of *appliances*. While very intriguing, I couldn't help but feel curious about whether the claims were accurate. So, after the conference I invited Eugene to come to our lab in Research Triangle Park in North Carolina to run some tests.

I have to admit now that in the back of my mind, I operated on the principle of "keeping your friends close and your enemies closer." Behind my intrigue was a feeling of wanting to understand their capabilities so that we could outperform vendors with WebSphere® Application Server. The tests went well; however, the DataPower team was somewhat reluctant to dwell on the raw XML performance capabilities of their appliance. Feeling a little suspicious, I had my team run some raw performance experiments. The results were off the charts. Why wasn't the DataPower team flaunting this capability? This is when I had my "ah-ha" moment. While performance measured in transactions per second is important and part of the value equation, the overall performance metrics found while assessing time to value and overall cost of operation and ownership are the most critical performance metrics to a business. This is where the DataPower appliances outperform. I read a paper, written by Jim Barton, CTO and co-founder of Tivo, called "Tivo-lution." The paper was inspiring as it confirmed the motivations and aspirations that I've had ever since I led IBM's acquisition of DataPower in 2005. In the paper, Barton describes the challenges of making complex systems usable and how "purpose-built" computer systems are one answer to the challenge:

> "One of the greatest challenges of designing a computer system is in making sure the system itself is 'invisible' to the user. The system should simply be a conduit to the desired result. There are many examples of such purpose-built systems, ranging from modern automobiles to mobile phones."

The concept of purpose-built systems is deeply engrained in our DNA at IBM. The name of our company implies this concept: International *Business Machines*.

IBM has a long history of building purposed machines, such as the 1933 Type 285, an electric bookkeeping and accounting machine. I can imagine this machine being delivered to an accountant, plugging it in, immediately followed by number crunching. The accountant didn't have to worry about hard drive capacity, operating system levels, compatibility between middleware vendors, or application functionality. It just did the job. I can also imagine it followed the 80/20 rule. It probably didn't do 100% of what all accountants needed. But it probably did 80% of what all accountants needed very well. Users just dealt with the remaining 20%, or learned to live without it.

"Business Machines, Again" is my inspiration. Our customers respond positively to the re-emergence of this approach to engineering products. It's all about time-to-value and total cost of operation and ownership. Appliances such as our WebSphere DataPower XI50 are leading the way in delivering on these attributes.

At the extreme, purpose-built systems, such as a Tivo DVR and a XI50, are built from the ground up for their purposes. While they might use off-the-shelf parts, such as an embedded Linux® OS, it is important that all parts are "right sized" for the job. Right-sizing source code in a hardware appliance is more like firmware (with strong affinity to the underlying hardware) than it is software. As such, the Tivo-lution paper describes the need to own every line of source code to ensure the highest level of integration and quality:

> "…by having control of each and every line of source code...
>
> Tivo would have full control of product quality and development schedules. When the big bug hunt occurred, as it always does, we needed the ability to follow every lead, understand every path, and track every problem down to its source."

The Tivo team even modified the GNU C++ compiler to eliminate the use of exceptions (which generate a lot of code that is seldom used) in favor of rigid checking of return code usage in the firmware. DataPower similarly contains a custom XML compiler that generates standard executable code for its general-purpose CPUs, as well as custom code for the (XG4) XML coprocessor card.

A physical appliance has the unparalleled benefit of being hardened for security. Jim talks about this in his Tivo paper:

> "Security must be fundamental to the design...We wanted to make it as difficult as possible, within the economics of the DVR platform, to corrupt the security of any particular DVR."

The DataPower team has taught me the meaning of "tamper-proof" appliances, or more precisely "tamper-evident." Like the 1982 Tylenol scare, we can't stop you from opening the box, but we can protect you, if someone does open it. In fact, the physical security characteristics of the DataPower XS40 make it one of the only technologies some of our most stringent customers will put on their network Demilitarized Zone (DMZ). If a DataPower box is compromised and opened, it basically stops working. An encrypted flash drive makes any configuration data, including security keys, difficult to exploit. "DP is like the roach motel; private keys go in, but never come out" is the way we sometimes describe the tamper-proof qualities of the XS40.

But the truth is, DataPower is not a DVR. DataPower is a middleware appliance. Middleware is a tricky thing to make an appliance out of. Middleware is enabling technology and by its nature is not specific to any application or vendor. The Tivo appliance is a specific application (TV and guide) that makes it somewhat easier to constrain:

> "Remember, it's television. Everybody knows how television works."
>
> "Television never stops, even when you turn off the TV set. Televisions never crash."

Hence, the challenge (and the art) in building a middleware appliance involves providing the right amount of constraint, without rendering the appliance useless. For example, DataPower does not run Java™ code (which is the primary means of customizing much of the WebSphere portfolio); instead, it uses XML as the primary mode of behavior customization. So, at some level, DP is not programmed, but instead it is configured. Now, for those who have used XML (and its cousin XSLT), you know that it's more than configuration; however, it is a constraint over Java programming, which has unbounded levels of customizability. The new combined team of IBM and DataPower have been bridging this gap (of special to general purpose) effectively. We have recently added features to DP to allow it to seamlessly connect to IBM mainframe software (IMS™ and DB2®) as well as capabilities to manage a collection of appliances as if they were one.

IBM has a healthy general-purpose software business. Our WebSphere, Java-based middleware is the poster child for general-purpose middleware (write once, run almost everywhere). However, there is a place for business machines that are purposed built and focus on providing the 80 part of the 80/20 rule. We are heading down this path in a Big Blue way.

This book represents an important milestone in the adoption of DataPower into the IBM family. The authors of this book represent some of IBM's most skilled practitioners of Service Oriented Architecture (SOA). This team is a customer facing team and has a great deal of experience in helping our customers quickly realize value from our products. They have also been among the most passionate within IBM of adopting the appliance approach to rapidly illustrating the value of SOA to our customers. The authors have unparalleled experience in using DataPower to solve some of our customers' most stringent systems integration problems. This book captures their experiences and best practices and is a valuable tool for deriving the most out of your WebSphere DataPower appliance.

—Jerry Cuomo, IBM Fellow, WebSphere CTO

Foreword by Kyle Brown

I can still remember the day in late 2005 when Jerry Cuomo first called me into his office to tell me about an acquisition (then pending) of a small Massachusetts company that manufactured hardware devices.

"Wait a minute. *Hardware??!?"*

That's the first incredulous thought that went through my mind. Jerry was the CTO of the WebSphere brand in IBM, which had become the industry-leading brand of middleware based on Java. Why were we looking at a company that made hardware? Echoing the immortal words of Dr. "Bones" McCoy from the classic Star Trek series, I then thought,

"I'm a software engineer, not a hardware engineer, dang it!"

But as I sat in his office, Jerry wove me a story (as he had for our executives) that soon had me convinced that this acquisition did, in fact, make sense for WebSphere as a brand and for IBM as a whole. Jerry had the vision of a whole new way of looking at SOA middleware—a vision that encompassed efficient, special-purpose appliances that could be used to build many of the parts of an SOA. Key to this vision was the acquisition of DataPower, which gave us not only a wealth of smart people with deep experience in Networking, XML, and SOA, but an entry into this field with the DataPower family of appliances—notably the XI50 Integration appliance.

Since that day, I've never regretted our decision to branch out the WebSphere brand well beyond its Java roots. The market response to the introduction of the DataPower appliances to the brand has been nothing short of phenomenal. Far from distracting us, the ability to provide our customers with an easy-to-use, easy-to-install, and remarkably efficient hardware-based option for their ESB and security needs has turned out to be an asset that created synergy with our other product lines and made the brand stronger as a whole. It's been an incredible journey, and as we begin to bring out new appliances in the DataPower line, we're only now beginning to see the fundamental shift in thinking that appliance-based approaches can give us.

On this journey, I've been accompanied by a fantastic group of people—some who came to us through the DataPower acquisition and some who were already part of the WebSphere family—who have helped our customers make use of these new technologies. Bill, John, and the rest of the author team are the true experts in this technology, and their expertise and experience show in this book.

This book provides a wealth of practical information for people who are either novices with the DataPower appliances, or who want to learn how to get the most from their appliances. It provides comprehensive coverage of all the topics that are necessary to master the DataPower appliance, from basic networking and security concepts, through advanced configuration of the Appliance's features. It provides copious, detailed examples of how the features of the appliances work, and provides debugging help and tips for helping you determine how to make those examples (and your own projects) work. But what's most helpful about this book is the way in which the team has given you not just an explanation of *how* you would use each feature, but also *why* the features are built the way they are. Understanding the thinking behind the approaches

taken is an enormous help in fully mastering these appliances. The team provides that, and provides you with a wealth of hints, tips, and time-saving advice not just for using and configuring devices, but also for how to structure your work with the devices.

This book is something the DataPower community has needed for a long time, and I'm glad that the authors have now provided it to the community. So sit back, crack open the book, open up the admin console (unless you have yet to take the appliance out of the box—the book will help you there, too!) and begin. Your work with the appliances is about to get a whole lot easier, more comprehensible, and enjoyable as well.

—Kyle Brown, Distinguished Engineer, IBM Software Services and Support

CHAPTER 1

An Introduction to DataPower SOA Appliances

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] 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.

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.

[1] 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, "DataPower Evolution."

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 encryption/digital signature capabilities as the other models that we discuss. For these reasons, it generally sits behind the DMZ,[2] 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:

- Encryption/decryption utilizing purpose-built hardware for RSA operations
- Digital signature creation/verification
- 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

[2] 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.

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.

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] 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.

[3] 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.

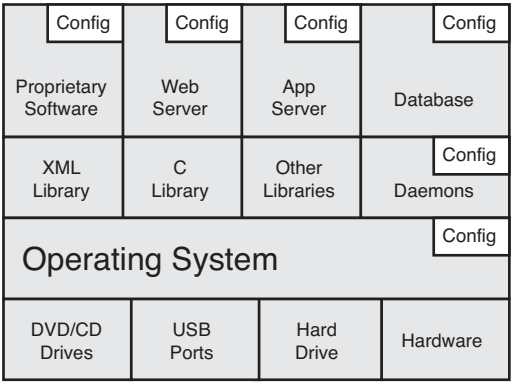

Figure 1-2 Typical server components.

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.

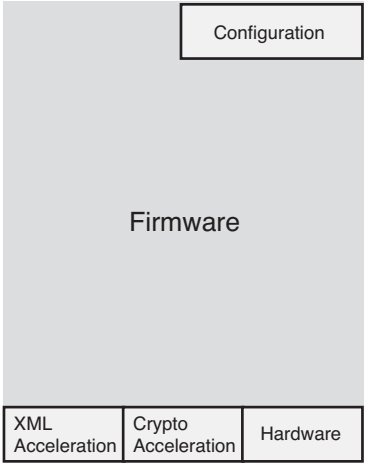

Figure 1-3 High-level SOA appliance components.

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* 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 "XML Threats," 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.

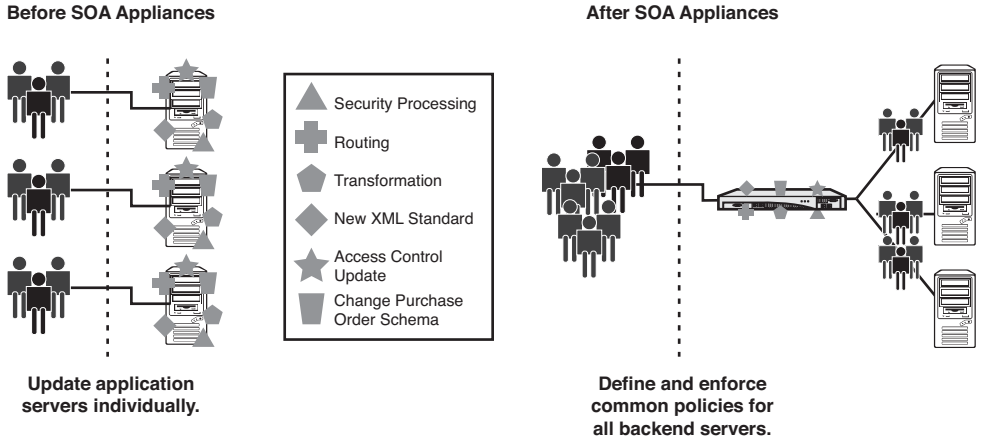

Figure 1-4 Redundant administration versus simplified appliance model.

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] for management.

[4] WSDM (Web Services Distributed Management) is a Web service standard for managing and monitoring the status of Web services.

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]

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.

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.

[5] 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.

Figure 1-5 Security impact of common tasks.

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 high-priority 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.

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, "Common DataPower Deployment Patterns," and Chapter 9, "Multi-Protocol Gateway."

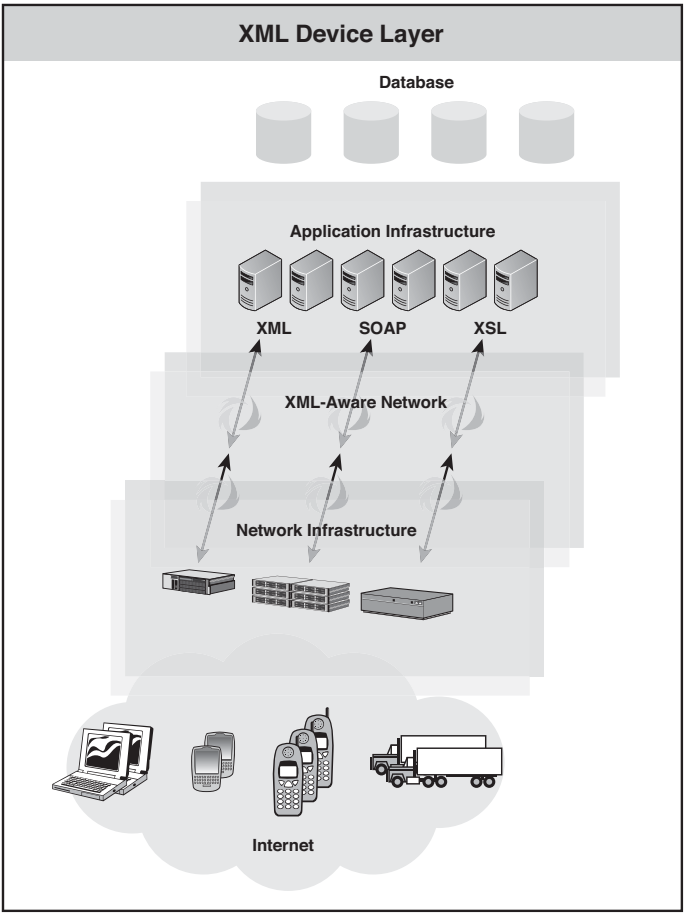

Figure 1-6 XML Aware Network layer.

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 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 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, "DataPower Quick Tour and Setup."

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] XPath allows for searching and retrieving information (nodesets) from XML documents based on some criteria.

[6] A newer and related XML specification named XQuery is much closer to true SQL capability.

- **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. After you do this, you have several options for administrative interfaces. These are described in detail in Chapter 12 "Device Administration" and Chapter 13 "Alternate Management Interfaces," 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] 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]

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

[7] This has in the past been referred to as SOMA, which is shorthand for SOap MAnagement.

[8] This is the admin interface for an XI50; the other models will have fewer features.

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.

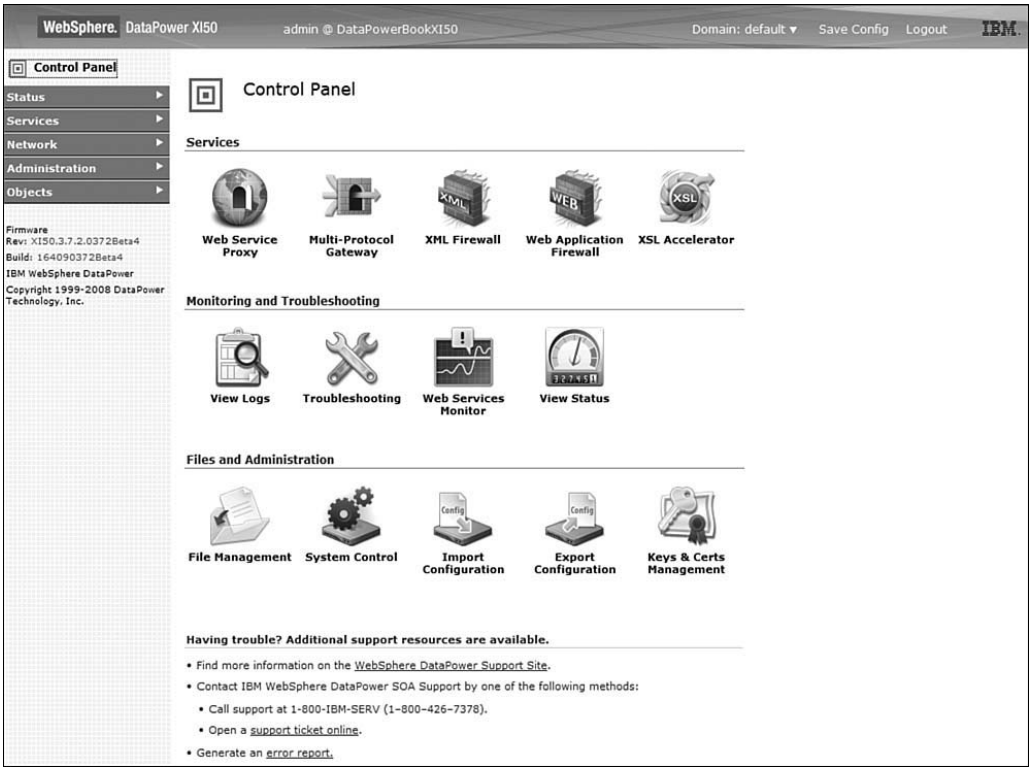

Figure 1-7 DataPower Web Admin console.

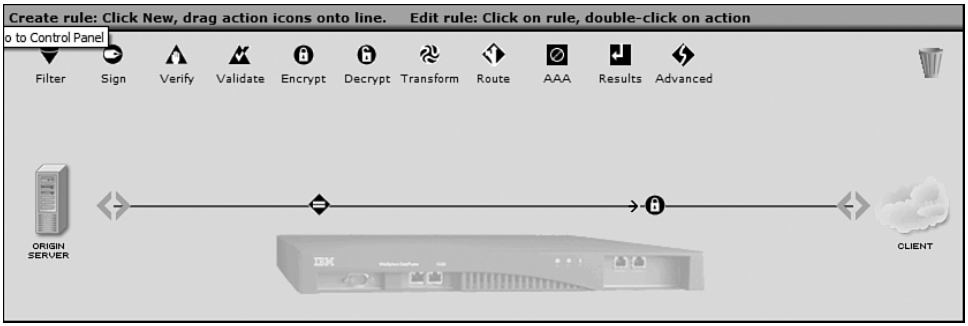

Figure 1-8 Drag-and-drop policy editor.

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, "Build and Deploy Techniques."

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, "DataPower Development."

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] 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.

[9] 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

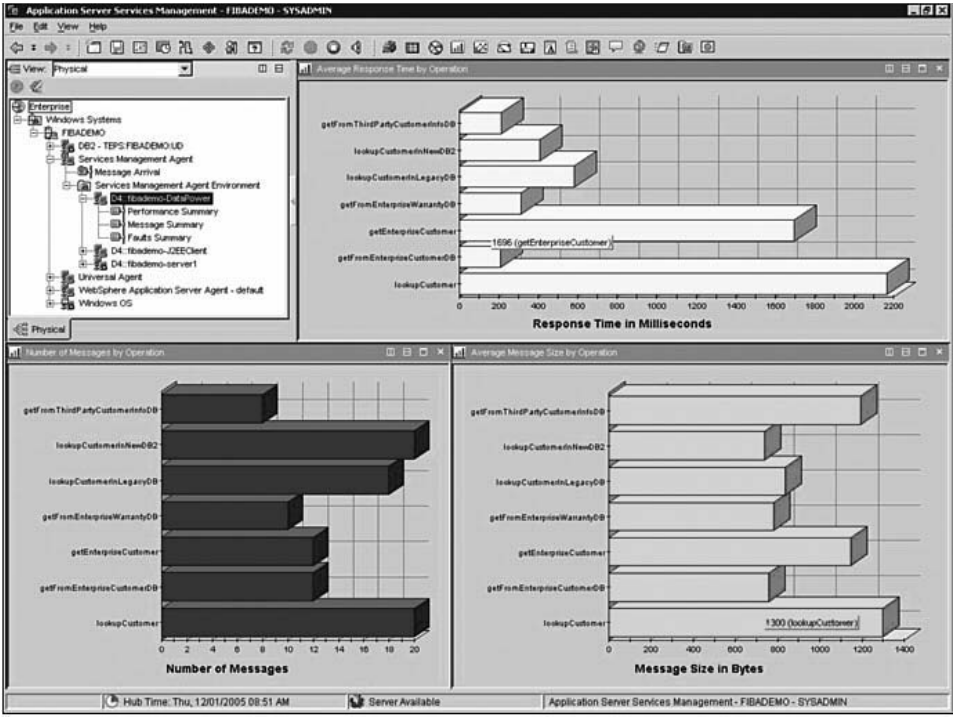

Figure 1-9 Monitoring DataPower appliances with Tivoli ITCAM for SOA.

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.

Included 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 Enterprise™ 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, "Multiple Device Management Tools."

Summary

This 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.

Index

B

C

D

E

F

G

H

I

J–K

L

M

N

P

Q

R

S

T

W

X–Z