# Chapter 5: Common DataPower Deployment Patterns

Add a note hereDataPower is often referred to by us in the field as a “Swiss Army knife.” You know—the red one with the white cross and a bazillion different blades and appendages. The analogy works because the knife has at least four or five blades capable of performing any given task required of it; some of them might do it faster, some require more brute force or cunning, but they’ll all result in the same vital task being satisfactorily completed. In the same way, there are almost always multiple ways that any specific task can be carried out using DataPower. Different services can be used to instigate the same processing—one might be easier to configure based on the inputs available (for instance the WSDL-defined Web Service Proxy [WSP] configuration), while another might be capable of reusing the same processing configuration for multiple inputs (the Multi-Protocol Gateway). Which one you choose depends not only on the task at hand, but on the context within which it is carried out.

Add a note hereThere are certain ways to do things that are generally seen as “always correct.” However, this doesn’t actually mean “always.” There is no such thing as a “best” practice, because each and every deployment is different, and there cannot be a single answer that is the best for everyone! There are, however, certain ways of doing things that have been done time and time again, and each time they are done, they have worked because they make sense. In IT, we refer to these as “patterns”—recurring deployment scenarios that can be reused in similar circumstances to give a similar positive result. This chapter describes some common DataPower deployment patterns—things that have worked well for other DataPower customers, and most likely will work just as well for you!

**Add a note here****Deployment Topologies**

Add a note hereA common DataPower misconception is that appliances have a single use or purpose. This might be true for your common garden-variety kitchen appliances but certainly not for DataPower! Whereas a dishwasher is only good for one thing—washing dishes—a DataPower appliance is adaptable and flexible. If it were a household appliance, it would wash your dishes, launder your clothes, vacuum your carpets, and mow your lawn—and it would do it as well, if not better than your existing appliances!

Add a note hereOkay, DataPower isn’t really going to metamorphose into a robot and go trimming your hedges (feel free to submit an Enhancement Request). However, note that each of the tasks mentioned has a completely different setting, depending on the appliance. It would require a single appliance that is just as good in the kitchen as in the garden, in the utility room as in the bedroom. In each area, there are different specific tasks that need to be carried out and carried out well. Imagine if they could all really be carried out by a single appliance that could be easily repurposed or reconfigured to perform whatever task was at hand.

Add a note hereDataPower is an amazingly flexible combination of hardware and firmware in a consumable form that is able to solve many of the extremely difficult SOA problems. It is precisely because of this flexibility that DataPower appliances are often deployed in completely different parts of an enterprise, performing completely separate roles. Deployments cover everything from security protection and application level firewalling to providing interoperability between disparate and incompatible systems to mediation and routing of messages as an Enterprise Service Bus (ESB). These are difficult challenges in the world of SOA that can be solved using a single configurable appliance instead of multiple complex software products that are difficult and expensive to install, deploy, and maintain.

Add a note hereThis section describes some of the most common deployment scenarios, although an infinite number of other possibilities exist where DataPower may well be a good fit.

**Add a note here****DataPower as Security Gateway**

Add a note hereOften the first response that comes to mind when someone asks “so what is this DataPower thing anyway?” is that it is an XML Security Gateway. That term somehow feels like it should have an ® or a TM somewhere behind it—that it is somehow official, that XML Security Gateway has a deeper meaning and standard behind it. It sounds as official as Firewall, Application Server or Database, and yet only a few years ago no one had heard of the term XML Security Gateway!

Add a note hereSo, what is an XML Security Gateway? It is a security hardened proxy that sits at the perimeter of your environment, usually in a network-level DMZ, terminates incoming connections, ensures that the requests are “safe,” and passes them on into your infrastructure. An example of an XML Security Gateway deployment pattern is depicted in Figure 5-1.

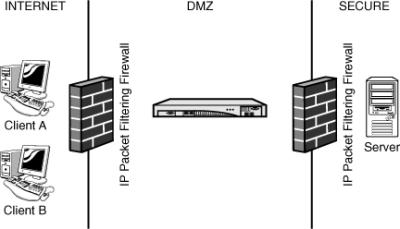
[](javascript:PopImage('IMG_67','http://images.books24x7.com/bookimages/id_30903/05fig01.jpg','496','104'))  
Add a note hereFigure 5-1: The XML Security Gateway.

Add a note hereFigure 5-1 shows a SOAP request to a server passing straight through IP packet filtering firewalls because those do not work on the application layer at all; this connection can be protected only by using an XML Security Gateway.

Add a note hereAs described in [Chapter 1](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=13#13), [“An Introduction to DataPower SOA Appliances,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=13#13) DataPower is a perfect fit for this type of deployment. The device is designed from the ground up for security and can be deployed into the DMZ without qualms. Moreover, it can ensure that requests are “safe” at multiple levels, including parsing the XML to ensure that there is no malicious or accidentally dangerous content for backend software XML parsers (discussed 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)) calling out to external virus scanners and other content checkers via the ICAP protocol (also discussed in [Chapter 20](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3695#3695)) and authenticating, authorizing, and auditing the requests (discussed in [Chapter 16](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2926#2926), [“AAA,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2926#2926) and [Chapter 17](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3157#3157), [“Advanced AAA”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3157#3157)).

Add a note hereThis level of protection is unprecedented in security devices. It deals with much more than the packet layer and the protocol layer; it is up at the application layer, OSI levels 5 through 7, dealing with message content. By actually parsing and processing the XML content, DataPower in the role of XML Security Gateway can protect against a class of real and increasingly sophisticated attacks against which to date there has been no real protection. The DataPower XS40 and the DataPower XI50 include functionality to act as an advanced XML Security Gateway.

Add a note hereA typical deployment scenario for DataPower as an XML Security Gateway is shown in Figure 5-2.

[](javascript:PopImage('IMG_68','http://images.books24x7.com/bookimages/id_30903/05fig02.jpg','432','248'))  
Add a note hereFigure 5-2: DataPower as the XML Security Gateway.

Add a note hereIn this instance, the role of the XML Security Gateway is being fulfilled by the DataPower XS40 appliance. The XI50, being a complete superset of the XS40, would also be able to carry out the role of XML Security Gateway.

Add a note hereIt’s all well talking in abstract terms about generic deployment topologies, but simply saying that DataPower can perform all these roles does not actually demonstrate that it is so. Let’s take a slightly deeper look at some of these functions and answer the question: What role does DataPower actually perform when deployed in each of these situations?

**SSL Termination**

Add a note hereThe decision on where to terminate SSL connections is an important decision for any Internetworked application environment. The SSL endpoint must be a secure hardened box with the capability to process the SSL handshake without detriment to performance and the capability to proxy incoming connections to backend servers. DataPower uses dedicated cryptographic hardware and can do this in an advanced manner, including handling client certificate authentication and specifying which SSL signers to trust, processing all the attributes in an SSL certificate to identify the owner, checking Certificate Revocation Lists and using the Online Certificate Status Protocol, providing specific server certificates depending on the requested IP address and port, specifying which encryption algorithms to use, and more—see [Chapter 18](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3288#3288), [“DataPower and SSL,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3288#3288) for details.

Add a note hereSSL termination on DataPower is shown in Figure 5-3.

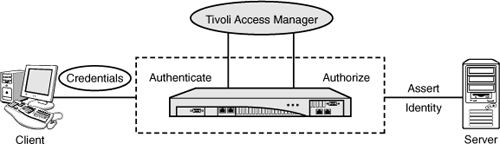
[](javascript:PopImage('IMG_69','http://images.books24x7.com/bookimages/id_30903/05fig03_alt.jpg','526','100'))  
Add a note hereFigure 5-3: DataPower can act as the SSL endpoint.

Add a note hereThe inbound HTTPS connection from the client is terminated at DataPower. Messages are then processed on the device, and when an explicit decision is made to do so in a Processing Policy, the messages are sent on to the backend server. Note that the connection to the backend is also over HTTPS—indeed this would likely be mutually authenticated SSL; however, this is a completely separate isolated and independent SSL session to that of the client connection.

Add a note hereEven when SSL is not in use, DataPower still acts as a connection termination point, ensuring that messages can never pass by without being inspected and deliberately routed onward.

**Authentication and Authorization**

Add a note herePerimeter security is an increasingly common requirement for complex customer environments. DataPower can authenticate credentials for incoming connections and make authorization decisions based on those credentials and the resources requested. This can be done in many ways, including out of the box integration with big-name enterprise authentication and authorization services and using many different standards. Figure 5-4 shows an example of how DataPower can integrate with an external provider for authentication and authorization.

[](javascript:PopImage('IMG_70','http://images.books24x7.com/bookimages/id_30903/05fig04_alt.jpg','577','166'))  
Add a note hereFigure 5-4: Externalized authentication and authorization.

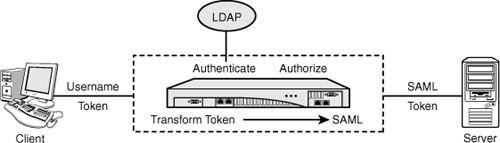
Add a note hereMore information on externalizing authentication and authorization can be found in [Chapters 16](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=2926#2926) and [17](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3157#3157).

**Token Transformation**

Add a note hereWhen Web service consumers authenticate, they do so in a myriad of different ways. There are many different types of tokens in the WS-Security standards, including a custom binary token which might be anything, and then even more possibilities beyond the standards. But what do we do if our Web service consumer wants to provide one type of token, but our Web service provider can only accept another type? Enter DataPower.

Add a note hereDataPower, sometimes in conjunction with another token transformation engine such as Tivoli Federated Identity Manager or other external authentication services, has the capability to mediate in the communication between consumer and provider to map tokens from one kind to the other. The number of types of tokens supported out of the box is impressive, and combined with the advanced cryptographic functionality on the device and the options for customization, the device can truly claim to be able to transform almost any kind of token into any other.

Add a note hereFigure 5-5 shows an example of one possible use case of DataPower performing token transformation.

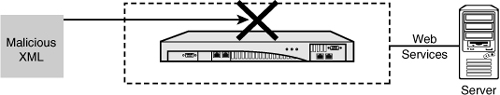
[](javascript:PopImage('IMG_71','http://images.books24x7.com/bookimages/id_30903/05fig05_alt.jpg','561','160'))  
Add a note hereFigure 5-5: Token transformation.

Add a note hereThe client sends a request containing a WS-Security Username Token (UNT), however the backend server with which they want to communicate understands only the Security Assertion Markup Language (SAML). DataPower validates the UNT against an LDAP directory, performs an authorization check on the request, and then transforms the token into a SAML authentication assertion—such that the backend server can understand it.

**XML Threat Protection**

Add a note hereIt’s a little known fact that even the most modern software XML parsers can be tricked by cleverly but maliciously written well-formed XML into using up significant system resources leading to systems going down or even remote system compromise. The same symptoms can also be induced by simple mistakes in writing input documents or application coding. The problem is that, by their nature, software parsers have to use up a huge amount of resources simply to realize that they are under attack. After they begin to process the XML, it is too late; often the attack has already done its damage.

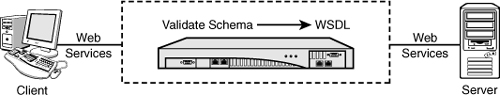
Add a note hereThis is where an XML Security Gateway is required. The security gateway proxies the backend service and processes the XML beforehand, barring malicious or badly written XML from entering the system, as shown in Figure 5-6.

[](javascript:PopImage('IMG_72','http://images.books24x7.com/bookimages/id_30903/05fig06_alt.jpg','535','102'))  
Add a note hereFigure 5-6: Malicious XML can lead to system compromise.

Add a note hereThe XML parser in DataPower, described in [Chapter 20](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=3695#3695), is different to these software XML parsers. It is constructed of dedicated custom hardware and a custom written firmware component, which has inherent safeguards built in. It can thus recognize more easily when there is something wrong and prevent such data from reaching the software-based XML parsers in your application servers behind it.

**Schema Validation**

Add a note hereSchema validation, with a well-written schema that does not use shortcuts like xs:any (a way of loosely typing data that makes it much easier to pass through malicious attacks) is a good way of verifying that the input into a Web service request conforms to expected parameters. Unfortunately, in too many cases, schema validation on software-based application servers is disabled, because of the simple fact that performing schema validation is an expensive operation in terms of latency and processing power. Figure 5-7 shows DataPower performing schema validation before allowing a request to continue to an application server, where it is now no longer required and would be significantly more expensive to do.

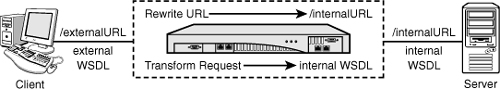
[](javascript:PopImage('IMG_73','http://images.books24x7.com/bookimages/id_30903/05fig07_alt.jpg','533','101'))  
Add a note hereFigure 5-7: Schema validation can be performed inline on DataPower.

Add a note hereDataPower as an XML Security Gateway provides the capability to do lightning-fast schema validation with minimal latency, using a unique patented XML parser that can validate the XML schema while the message streams through the device. This means that there is now no excuse not to perform schema validation on inbound Web services requests.

**Resource Masking**

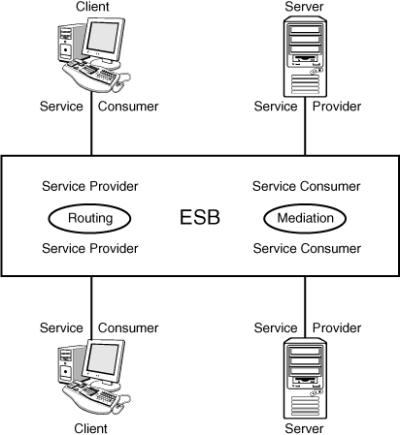
Add a note hereWhen people publish business services on the Web as Web services, they often do not consider what exactly it is they are making available. Thus, a Web service that is automatically generated from some business logic code using a tool designed to simply make processing methods available may actually be giving external users more than was intended. For instance, if a piece of business code provides methods to add and remove money and view the amount that was in an account, but only the view method was intended for external consumption, clearly it is only that view method that should be visible and accessible from the outside. Internal resources can also include URLs and extra parameters on requests; these should not be made visible to external callers where there is no need to do so.

Add a note hereDataPower has facilities to assist with this resource masking. It can rewrite URLs to only allow through specific methods, to change the method names requested, to modify the message format in transit, and in general, to make the frontend service look nothing like that at the backend. Figure 5-8 shows how a resource might be masked using DataPower.

[](javascript:PopImage('IMG_74','http://images.books24x7.com/bookimages/id_30903/05fig08_alt.jpg','542','96'))  
Add a note hereFigure 5-8: DataPower resource masking.

**Add a note here****DataPower as Enterprise Service Bus**

Add a note hereThe Enterprise Service Bus pattern has been described in many books and papers. It is an architectural style, a way of designing intermediary computer systems to process and route data to appropriate backend systems or endpoints in an efficient and secure manner. The ESB pattern is all about service connectivity; in ESB parlance instead of clients and servers, we talk of service consumers and service providers. The ESB acts as a service provider to service consumers and as a service consumer to service providers. Figure 5-9 shows the ESB pattern at a high level.

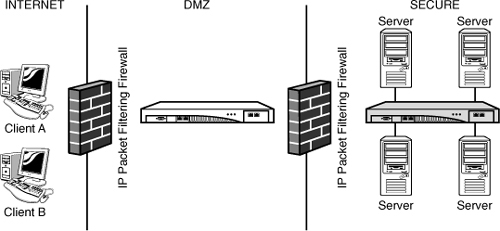
[](javascript:PopImage('IMG_75','http://images.books24x7.com/bookimages/id_30903/05fig09.jpg','412','449'))  
Add a note hereFigure 5-9: The Enterprise Service Bus pattern.

Add a note hereTwo fundamental concepts underlie the ESB style: routing and mediation. Routing in an ESB is much like network routing in concept; a service consumer connects to the bus, and the infrastructure decides which service provider to route the request to. Of course, because the ESB is acting as a service provider to that consumer, it can transparently route the request to any real provider of the service that it wants. It could also respond to the consumer itself, without ever contacting a service provider; it can even contact a number of service providers, aggregate the results, and return them as a single response to the service consumer. Mediation is a slightly more advanced concept that comes into play when service consumers and service providers do not match, when they either do not speak the same wire format, data format, schema, or any other kind of difference. Mediation bridges the differences by transforming data formats, converting transport layers, remapping data structures, and repurposing existing services such that they can fulfill others.

Add a note hereFinally, as an ESB performs its mediation and routing of service requests from consumers to producers and back, it by design adds more value in the form of security enforcement, service-level management and monitoring. A well-designed ESB can perform these functions not just on services or specific providers but on as fine-grained a level as required, even to sections of the individual invocations of specific services. There is a lot of power inherent in the ESB architectural style!

Add a note hereDataPower is a perfect implementation platform for an Enterprise Service Bus, to the extent that IBM has explicitly included it in their ESB strategy and christened the DataPower XI50 appliance the “Hardware ESB.” Implementations of routing and mediation are configured, often using processing policies created from out of the box processing actions, to determine appropriate service providers to service specific requests. Where the device shines is in the implementation of the added value features. Performing security enforcement, token transformation, service-level management and monitoring are the bread-and-butter of the appliance, as will become apparent throughout the later chapters of this book.

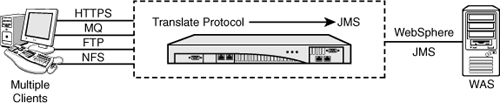
Add a note hereFigure 5-10 shows a DataPower XI50 deployed as an ESB, in the secure zone and behind another appliance (in this case an XS40) acting as an XML Security Gateway. As discussed in [Chapter 4](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=403#403), [“Advanced DataPower Networking,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=403#403) the functions could both be served from the same device if security rules on the network are allowed, but often there are hard and fast rules that mandate separate physical devices for these functions.

[](javascript:PopImage('IMG_76','http://images.books24x7.com/bookimages/id_30903/05fig10_alt.jpg','539','249'))  
Add a note hereFigure 5-10: DataPower deployed as an ESB.

Add a note hereAs an ESB, there are certain “sweet spots”—things that clearly demonstrate that you made the right choice in having DataPower be part of your ESB implementation. This section explores some of those use cases and the role that DataPower plays in them.

**Multiple Protocols**

Add a note hereOne of the chief advantages of DataPower is that it is capable of communicating with many different technologies. Its protocol support covers a number of currently popular transport protocols, including HTTP/HTTPS, FTP/FTPS, WebSphere MQ, NFS, IMS, WebSphere JMS, and Tibco EMS. As will become obvious throughout the book, the device is at home with any of these protocols, and they allow it to communicate with almost any modern Internet-connected system. Figure 5-11 shows an example of DataPower processing multiple protocols.

[](javascript:PopImage('IMG_77','http://images.books24x7.com/bookimages/id_30903/05fig11_alt.jpg','575','118'))  
Add a note hereFigure 5-11: DataPower can process multiple protocols.

Add a note hereHTTP and HTTPS are, of course, used for the most-often discussed protocol of the Enterprise Service Bus—Web services. But they are also used for many other forms of communication. Sometimes integration of “traditional” Web applications, designed not for remote system use but for human interaction, is done over HTTP, with the system or bus pretending to be a human user. And, other patterns of stateless or stateful system interaction over HTTP, such as RESTful services, are becoming more and more commonplace. A good ESB must be able to support these.

Add a note hereFTP support is absolutely crucial. There are many “legacy” systems in existence whose main form of communication is files. That is, they accept input files that are put on a file system, and they produce as output other files that are put back onto the file system. To communicate with these systems, we must have a way of processing data from that remote file system; with FTP support, this becomes trivial, because almost every IP-connected system can easily be configured to communicate via FTP! However simply supporting the protocol is not enough; we must be able to support it securely, such that the administrators and owners of these legacy systems are happy allowing us to access them. Thus the DataPower implementation of FTP allows for full AAA processing and for configuration of mutually authenticated SSL over both the control and data channels as required.

Add a note hereLikewise, Network File System (NFS) support is key to integrating with systems that are not quite “up to date” enough to support Web services. Large shared Network Attached Storage (NAS) devices are becoming common in the enterprise and provide a shared platform over which data can be exchanged; DataPower is able to poll and process files from these NAS arrays using NFS and write files to them in response. In addition, DataPower provides an iSCSI Host Bus Adapter (HBA) on some appliances, which can communicate using the iSCSI protocol with iSCSI targets.

Add a note hereFinally, the messaging protocols: WebSphere MQ, WebSphere JMS, and Tibco EMS. Ever since the advent of the IBM Message Queuing protocol, asynchronous queuing has been hugely popular for two reasons. The first is that it provides an asynchronous method of submitting data for processing; that is, you write a message to a queue, and something else will deal with it in its own time. This is a great way of processing large loads at peak efficiency because business systems can draw exactly enough data to process at a time and are short of data to process only if there is no actual work to do; for this reason alone it is vital that an ESB support asynchronous messaging. However there is another reason that applies to WebSphere MQ, and this is that MQ client and server software support is available for an amazing array of platforms. This means that we can use MQ to communicate with everything from a Windows server to a mainframe. Unfortunately, this also often leads to designs that are “less than optimal” from an architectural point of view, where the asynchronous queuing mechanism is used synchronously—clients put a message to one queue and wait for a response on another queue before continuing their processing—but this is of less importance than the fact that they can use the protocol to communicate with a system with which they would otherwise be unable to communicate!

Add a note hereTo implement an effective ESB, we must be able to communicate as widely as possible with as many different disparate systems as possible, such that those systems can become consumers of the services we provide and providers of the services we consume. DataPower’s protocol support is wide-reaching and covers the core protocols that are popular today.

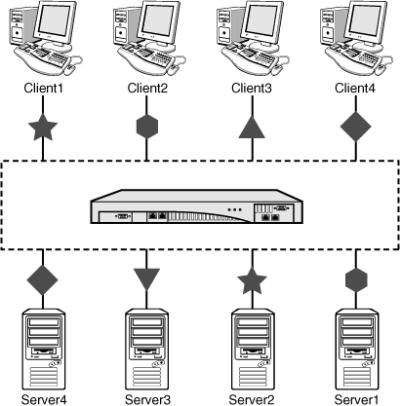
**Multiple Datatypes**

Add a note hereJust as we can communicate with many different protocols, the DataPower XI50 is actually capable of processing almost any type of data. What’s that I hear you say? Isn’t it an XML appliance? Well, yes, it is—much of the internal workings of the appliances is XSLT, and it has hardware XML acceleration to ensure that the performance cost of processing XML at high speed is not an issue. But the creators of DataPower realized that not everything in the world is XML, and that there are a vast amount of systems out there that are not and likely never will be able to communicate using XML. Thus they designed in the capability to work with non-XML data.

Add a note here[Chapter 26](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4607#4607), [“Transforming Non-XML Data,”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4607#4607) describes in detail how this functionality can work with non-XML formats—indeed there is a specific example of communicating with a backend using a COBOL Copybook. This extremely powerful functionality in conjunction with the support for different communication protocols means that DataPower is capable of performing integration that in some cases would simply not be possible in any other way without writing significant amounts of application code—so much in fact that it would probably be simpler to rewrite the backend systems! That is the value proposition: Really difficult SOA core work can be completed more easily and mainly using configuration on DataPower.

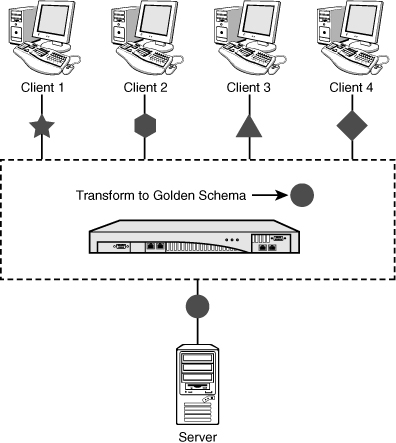
Add a note hereIn addition, it is important not to forget that one of the fundamental core functions of the device is transformation of XML data. XSLT allows for structured and controlled transformations and translations of formats that have absolutely nothing in common with each other except for the fact that they are both expressed in XML.

Add a note hereThis ability to perform many transformations is important when dealing with real-life use cases. In the real world, nothing is as simple as it should be. For example, when a system provides a service that is needed, and other systems want to use that service, there is no guarantee that those other systems will in any way be compatible. A typical enterprise might have any number of different clients all performing the same kind of processing, and all doing it with their own definition of what that processing means. Each client and each server may have its own schema and its own understanding of how to process the data, and therefore, the service provider must understand and implement all of them. Figure 5-12 shows this interaction.

[](javascript:PopImage('IMG_78','http://images.books24x7.com/bookimages/id_30903/05fig12.jpg','410','417'))  
Add a note hereFigure 5-12: Many clients, many servers, many schemas.

Add a note hereOutside of the enterprise there are also many situations in which people do similar things in a similar way, but none of the ways they do things can interoperate with each other. If you look at any given industry, it is highly likely that they will have common industry concepts that anyone performing that kind of business must be able to understand; otherwise, they would be out of business. The more established industries have, over time, defined common standards that everyone adheres to in order to be able to interoperate and communicate. However even those standards are often open to interpretation, or expanded on for commercial advantage, or worse yet competing standards arrive and need to be translated between! The best example of this is the original Web Services standards, which were so noninteroperable that they lead to the creation of the Web Services Interoperability Organization that had to create “Profiles” to explain how to make them interoperate!

Add a note hereUsing DataPower, it is possible to process the data for all these scenarios. But what do you do with it once you have it? A useful pattern is to create what is loosely known as a “golden schema.” This is a concept where you take all the common points of data and wrap them up into a single definition and representation of that data. Figure 5-13 shows mapping to a golden schema.

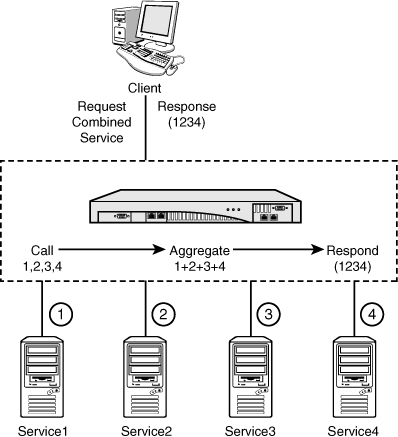
  
Add a note hereFigure 5-13: The golden schema.

Add a note hereThis is hard work, and DataPower will not magically come up with the schema for you! However DataPower provides the frontend to enable mapping of all the disparate data types and formats into this golden schema so that your backend applications can work with this single, united understanding. This leads to a far simpler development process for new applications and with time can be used to provide back to the clients (be they internal clients or industrywide) that such-and-such a definition might be useful because it truly represents the needs of the industry.

**Routing and Orchestration**

Add a note hereWe mentioned earlier that routing and mediation are the two core concepts of an ESB. DataPower is more than capable of implementing both of these; its capability to handle multiple datatypes makes it perfectly placed to mediate between applications. Routing, however, can be extended into a further paradigm known as orchestration.

Add a note hereThe primary goal of an ESB is to make it easier to integrate applications and services. Service routing entails taking a request from a service consumer and passing it on to a service provider that will be able to fulfill that request. DataPower can route messages in a myriad of different ways, and even contains an out-of-the-box dynamic [“routing”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=4345#4345) action that can be freely used in a Processing Policy. But what if, rather than simply passing through a request and passing back a response, while potentially transforming the data structure and changing the protocols, we were to take that request and provide a response by building up a number of requests to several services? Would that make DataPower the service provider or just another intermediary? This concept of brokering multiple services into a single combined composite service is known as *orchestration*, which is depicted in Figure 5-14.

  
Add a note hereFigure 5-14: Orchestration of multiple services.

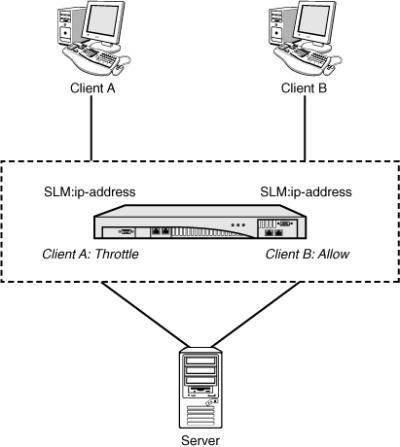
Add a note hereThe specific orchestration shown in Figure 5-14 is simple; it shows the calling of multiple services in order and aggregating the responses. DataPower’s capability to perform orchestration out of the box using simple actions has been extended with the ability to send out multiple asynchronous requests to backend servers and then aggregate the responses in this manner. When using custom processing for orchestration, the device shines because orchestration requires a detailed knowledge of the service to be composed and the services from which it will be formed.

**Add a note here****Web Services Management**

Add a note hereThe third deployment topology that proves more popular with DataPower customers is that of Web Services Management. This use case seems relatively simple at first, but is, in fact, multifaceted and has a lot of detail and complexity hidden within, making it as powerful and compelling a proposition as the first two.

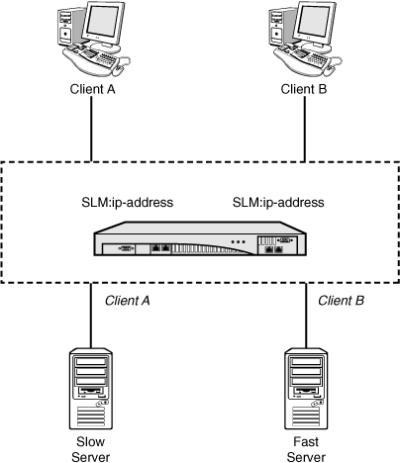
Add a note hereThe idea behind Web Services Management is that, for specific services, different consumers of those services may require different levels of service. For instance, if Client A is paying less for a service and Client B is paying more, and there is some kind of a problem that not all requests can be serviced, it seems that Client B, who is paying more for the service, should be given priority over Client A. Of course, Client A would have known up front when choosing the less expensive package that in the event of issues, other clients paying for more expensive packages would be prioritized over him.

Add a note hereHow would we implement this on DataPower? Well, DataPower provides a facility called Service Level Monitoring (SLM), which is explored in depth in [Chapter 10](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1548#1548), [“Web Service Proxy.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1548#1548) In short, the SLM capability allows for a policy-driven approach to decide with which priority requests are treated. It can throttle or even deny requests that meet certain preset boundaries and conditions, which can be dynamically reallocated at runtime. This is used to enforce policies such as that described previously with different clients paying different fees for the same service but with different Service Level Agreements (SLA), as shown in Figure 5-15.

[](javascript:PopImage('IMG_81','http://images.books24x7.com/bookimages/id_30903/05fig15.jpg','411','460'))  
Add a note hereFigure 5-15: Throttle Client A’s traffic because he paid less.

Add a note hereIn the scenario in Figure 5-15, Client A pays less for a service and Client B pays more for the same services. DataPower implements SLM based on the IP address of the client, and because it can identify the client, it can make a decision about whether to allow the traffic through.

Add a note hereSLM can be combined with other capabilities in DataPower to perform more advanced actions. For instance, combining SLM with dynamic routing would allow a variation of the pattern from Figure 5-15, shown in Figure 5-16.

[](javascript:PopImage('IMG_82','http://images.books24x7.com/bookimages/id_30903/05fig16.jpg','404','468'))  
Add a note hereFigure 5-16: Different service levels for different clients sent to different servers.

Add a note hereIn the scenario in Figure 5-16, there are two backend servers that are capable of providing the service: Slow Server and Fast Server. DataPower again implements SLM based on the IP address of the client (although this could, of course, be based on any of a vast selection of transactional metadata), and this time uses that information to direct requests from Client A to Slow Server and Client B to Fast Server, thus ensuring that Client B gets a better service.

Add a note hereThe SLM functionality can also be used to protect backend application servers from spikes in traffic. Where the capacity of a server or group of servers is known, it is possible to utilize SLM as a traffic management point where advanced traffic shaping and throttling occurs and requests are queued on the network or in the device up to a limit or until the application server is capable of processing it, or even simply rejected by configuration for specific clients until the traffic situation abates.

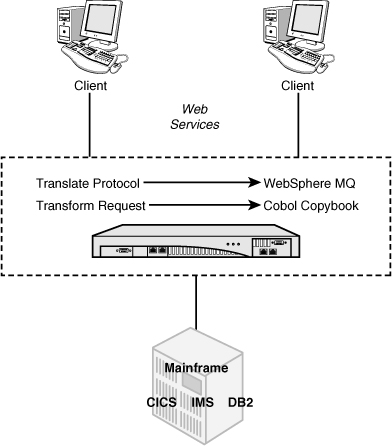
Add a note hereFinally, and perhaps most importantly, SLM allows for data to be collected on who is using which backend systems and how. This kind of data is invaluable to enterprises that provide services, because it can become the basis on which to provide usage reports, create cross-charges or chargeback for services used, and monitor the status of services.

**Add a note here****Web Services Enablement of Legacy Systems**

Add a note hereAnother important DataPower deployment topology is that of enabling legacy applications to work in the modern era of Web services. Legacy applications such as CICS® and IMS, usually running on mainframe computers that have been doing their jobs steadily and well for many years, are not significantly different in concept to our modern application servers. They are often designed in a similar idiom to modern form-based Web applications; the user types data into a number of fields, clicks the submit button, and the application processes the input and returns a response. However there are a number of challenges with modernizing these applications.

Add a note hereFirst of all and most important is the fact that, because these applications have been stable and running for many years, it is very hard in most environments to get a change made to the application. It is possible for implementation of changes to take months or even years! In some cases it is even all but impossible, because the source code to the applications has been lost over the years. Secondly, the applications often use older formats for data and requests, which are not well supported in the age of the Web browser. And finally, the applications are often not capable of communicating via the modern protocols currently in use.

Add a note hereDataPower is a wonderful solution to all these issues. Its support for protocol bridging and data transformation mean that it is possible to configure and front these “legacy” applications with a minimum of fuss, and significantly less effort than would be required with an application coding effort. Moreover it is usually possible to perform this integration without requiring any changes on the mainframe—often turning an impossible project into a realistic one. Figure 5-17 shows DataPower providing a Web services interface for a legacy application.

  
Add a note hereFigure 5-17: Web Service enabling CICS.

Add a note hereIn addition, there are extra benefits to consider. First, the legacy applications might not have data secured at a granular level of access control. When these applications were written, it was common practice to secure at the level of application access; that is, to provide an identity for a remote application which, after authentication, had access to do whatever it needed to the data. By leveraging DataPower’s powerful AAA functionality, we can add more fine-grained security around this data.

Add a note hereSecond, DataPower is a high-performance appliance. By processing more of the business logic on the appliance and integrating with the application only at the level of the data to be processed, and by pre-processing wherever possible, it is highly likely that less CPU (measured in MIPS) will be required on the mainframe applications. This may even lead to a reduction in cost for running those applications for some customers!

Add a note hereFinally, there is the advantage that, by exposing these legacy applications as reusable Web services on an ESB as part of an SOA, we can leverage the existing applications to perform their functions as part of wider processing of newer applications, thus extending the useful life of the legacy services.

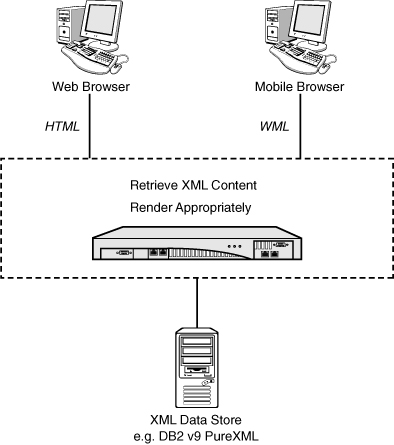
**Add a note here****Dynamic Content Rendering**

Add a note hereThe final deployment pattern we discuss is that of dynamic rendering of content. This may seem to be a simple use case, but it is powerful; IBM DataPower clients use the appliances to serve high-traffic Web sites to a multitude of different clients. Moreover, this scenario can be implemented on all the appliances: XA35, XS40, and XI50.

Add a note hereXSLT, the transformation language at the heart of DataPower, was designed to dynamically generate a markup language for the Web—HTML—and this is still a common usage. This usage becomes an order of magnitude more useful when combined with the concept of *transcoding*. If all our content is stored in render-neutral XML, and we create stylesheets to transform it into HTML, we can also write other stylesheets to transform the content into other markup languages, such as Wireless Markup Language (WML). The WML use case is especially interesting, because this is a common format for handheld wireless devices. We could thus dynamically choose whether to render our content for a normal Web browser or a mobile browser—without needing to explicitly modify the content itself.

Add a note herePerforming this translation on DataPower is highly performant as the device itself is capable of processing transformations at extremely high speeds. In addition, DataPower’s support for dynamic processing makes it easy for us to render the relevant markup depending on the type of device—for instance, we can recognize the device type based on HTTP headers and run the relevant stylesheet.

Add a note hereTo deploy DataPower for dynamic content rendering, it is important to ensure that the device is deployed inline in the processing—that is, clients request the content from DataPower, which then calls back to wherever that content is stored, retrieves it, transforms it, and returns the transformed content back to the requesting client. This pattern for deployment provides much higher performance than “calling out” to DataPower from within another processing flow, and is shown in Figure 5-18.

  
Add a note hereFigure 5-18: Deploying DataPower for dynamic content rendering.

## Summary

Add a note hereChoosing how to deploy DataPower is a complex decision process, because there are so many possible ways to do it, and so many places in your infrastructure where it is just a perfect fit to do the job! This chapter presented some common deployment patterns and some ways of deploying the appliance that are known to have worked for other customers and will likely work well for you.

Add a note hereThis also brings to a close [Part II](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=307#307), “[DataPower Networking](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=307" \l "307" \t "_parent).” [Part III](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=754#754), “[DataPower Services](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=754" \l "754" \t "_parent),” goes into more detail for how to configure the device to perform the wonderful things it can do. Onward!