**Chapter 6: Introduction to DataPower Services**

Add a note hereSo now that you’ve heard all the great things that the DataPower SOA appliance can do for you, you’re probably on the edge of your seat, ready to plug one in and watch it do its magic! Well, there is some assembly required after the appliance is plugged in to your network. Unlike your typical household appliances such as a refrigerator, where you simply plug it in and load all the food, these appliances need to be configured to do precisely what you want them to do. If they are not configured, they will not allow any traffic to pass through them. This is not by mistake, though. The devices are intentionally factory set secure by default, where no traffic is allowed through the device unless it is explicitly configured to accept it. As mentioned 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) it is much more secure to turn on everything you want rather than turn off everything you don’t want.

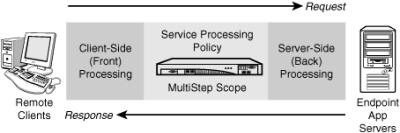
Add a note hereThe configurations that you create on the appliances to process and control traffic are called *services*. This chapter describes what a DataPower service is, what components combine to form one, and the different types of services available. After reading this chapter, you should have a general understanding of DataPower services.

**Add a note here****What Is a DataPower Service?**

Add a note hereDataPower SOA appliances can be configured to perform various types of processing on message requests and responses, such as security, transformation, mediation, and routing. In general, you create proxies for your applications to perform these actions before the messages are sent to them. The proxies created on the appliance to provide these functionalities are called services. For example, you might configure a service type called a Web Service Proxy on DataPower to front the Web service providers on your backend.

Add a note hereIn general, a DataPower service is a simple concept; however, it is critical to understand what a service is and how it works as it is the component that is configured to apply any and all processing to incoming and outgoing messages. Without a service configured to listen for and process incoming messages, no traffic can flow though the appliance, assuring that only desired traffic is accepted and providing maximum security.

Add a note hereIf you think of a service as a proxy, and understand what a proxy does, you can visualize that there are three high-level processing phases that must be performed for all requests and responses. A service accepts an incoming request on the established front-side connection, processes the request, and passes the request to the backend server over the backend connection. These same phases are also invoked in reverse order on the response from the server back to the client. Figure 6-1 illustrates the three phases of a DataPower service and the order in which they are invoked on a request and response message.

[](javascript:PopImage('IMG_85','http://images.books24x7.com/bookimages/id_30903/06fig01.jpg','454','152'))  
Add a note hereFigure 6-1: DataPower service processing phases.

**Add a note here****Client-Side (Front) Processing**

Add a note hereAs a request is submitted to the DataPower appliance, there could be many different services configured to process incoming messages; however, only one can receive a single request from the client, so there must be something in the configuration that decides which service will handle it. This is determined by the IP address and port combination that the service is configured to listen on. The one exception to this rule is when defining a Web Service Proxy where this can be defined at a more granular level. This will be discussed in detail 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) A service can also be configured to poll for messages, but that is covered in [Chapter 9](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1276#1276), “[Multi-Protocol Gateway](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=843#843).” For simplicity, we assume that it is listening on a specific port for incoming requests.

Add a note hereAfter a service has received the incoming message, there is additional client-side processing that will be applied. If the request is using SSL, this phase will perform the specified level of SSL negotiation and establish the SSL connection, which may include mutual authentication. Transport level decryption of the datastream is also performed for requests sent over SSL.

Add a note hereAfter the SSL connection has been established and the datastream has been decrypted (if applicable), this phase will apply additional processing on the request such as Service Level Monitoring (SLM), threat protection, attachment processing, message throttles, and URL rewriting. All this processing is configurable within a service and is discussed in [Chapter 7](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=868#868), “Introduction to Services Configuration.” For now, it is important to understand that there is a significant amount of processing that occurs in the client-side (front) processing phase before the service even begins to process the message. This phase could reject the message before any message processing is even attempted. Much of this is all that “good stuff” that DataPower provides out of the box such as schema validation and message integrity, while some of it needs to be configured. Again, don’t worry about how this all happens for now, just realize that this is where it happens.

**Add a note here****Service Processing Policy**

Add a note hereThis is where all the action is in a service (literally). This phase is also known as multistep processing. After a request to the service is deemed acceptable to the service by passing all threat protection and other client-side (front) processing as described previously, the service can start to perform all the configured processing operations on the message. This is known as the service’s Processing Policy. A Processing Policy is a list of rules that contain processing actions that can be applied to a message. Actions are specific processing operations that are applied to a message such as document format transformations, encryption and decryption, message signing, authentication, and so on. As the request message passes through the Processing Policy, the actions are applied to the message in the specified sequence, producing a final message to be passed to the next and final request phase of the service.

**Add a note here****Server-Side (Back) Processing**

Add a note hereAt this point, the request has been accepted and processed by the client-side (front) processing phase, processed by the Service Policy, where all the configured actions were applied and is now ready to be passed to the backend server. Before sending the final message directly to the backend server, there may be some additional steps required. This could include a new SSL connection to the backend server (remember, the frontend client SSL connection is always terminated at the device), setting additional headers in the request, and setting the protocol version that the backend server expects. For example, suppose the incoming request from the client uses HTTP version 1.1 and the backend is a legacy application that supports only HTTP version 1.0. The server-side (back) processing phase would send the request using HTTP version 1.0 as expected by the backend application.

Add a note hereOne last and important job of the server-side (back) processing phase is to forward the request to the backend server after it is done being processed by the service. This server location can be defined in one of two different ways. If all requests flowing through the service are always forwarded to the same server, a static backend can be defined in the service configuration. The second method is a powerful feature of a DataPower service that provides the ability to perform dynamic routing. This can be accomplished by defining the backend server as dynamic within the service configuration. It is then the responsibility of the Processing Policy to determine the server location at runtime. This decision can be based on metadata such as protocol headers, URI, or the message content itself.

**Add a note here****Response Processing**

Add a note hereTo this point, the processing phases described have focused on the request from the client to the server. All this processing can also occur on the response back from the server to the client. The backend connection is used to receive the response from the server, the Processing Policy is invoked for response message processing, and finally the frontend connection is used to forward the response to the client. The same types of processing can occur on the response as in the request. The service can be configured to perform the same exact processing as in the request or entirely different processing. For example, if the request is sent in a format that is unrecognizable to the backend application, the service can perform a transformation on the request document to convert it into a format that the backend application can recognize. As the response is sent back from the backend application, the service can perform a similar transformation on the response document that will convert it into a format that the client would recognize.

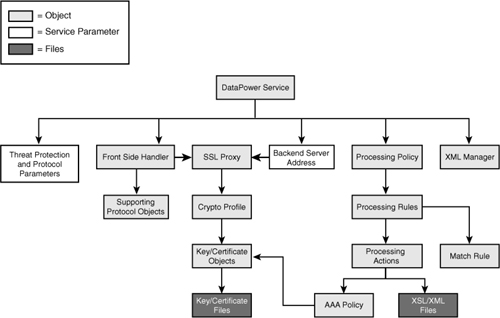
Add a note hereThe three phases described previously show a simplified, conceptual view of a DataPower service. Each of the three phases contains many configuration parameters and options. The available options and parameters may vary by service type. The various service types are discussed later in this chapter; however, the configuration details of the different service types are each covered in separate chapters.

**Tip: Pay Attention to the Sequence of Processing**

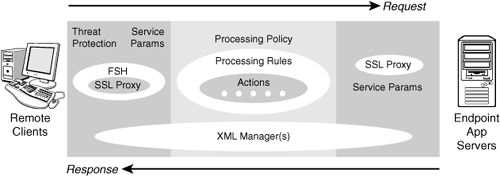
Add a note hereMany man-hours can be wasted looking to do or troubleshoot things in the wrong order.

**Anatomy of a DataPower Service**

Add a note hereBy now you should have a conceptual understanding of what a DataPower service is and what it does, and you’re probably ready to dive right in and start creating your services. Not so fast! Before you can create one of your own services, it is important to understand the components of a DataPower service. There are lots of moving parts in a service that can be created, configured, or referenced to provide the end-to-end processing. A DataPower service is object-oriented, so that almost every part of a service is an independent, reusable object. A DataPower service contains a collection of objects that are each configured to perform a specific task. These objects are arranged in a hierarchy with many references and dependencies. Many of these objects will be created automatically for you (depending on how you set up the service), but it is important to know about them even if you never configure them manually. Figure 6-2 shows a top-level object hierarchy of a DataPower service.

[](javascript:PopImage('IMG_86','http://images.books24x7.com/bookimages/id_30903/06fig02_alt.jpg','796','507'))  
Add a note hereFigure 6-2: DataPower service object hierarchy.

Add a note hereEach of the objects in DataPower services are invoked in one of the three processing phases discussed earlier in the chapter. Figure 6-3 shows the three processing phases, similar to [Figure 6-1](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=771#771), but also includes the main service objects in the phase that they are invoked in a request/response transaction.

[](javascript:PopImage('IMG_87','http://images.books24x7.com/bookimages/id_30903/06fig03_alt.jpg','631','223'))  
Add a note hereFigure 6-3: Object processing order.

Add a note hereThe following list provides a high-level description of each of the main service objects shown in Figure 6-3. A more detailed explanation and the configuration of each object are discussed in future chapters:

* Add a note here**Front Side Handler—** The Front Side Handler (FSH) object is the first point of entry into the DataPower service. It is kind of like the gatekeeper for your service. It is the object that is configured to listen on a specific IP and port combination for incoming requests (or poll for messages) and perform some level of validation on the request. Most of the validation being performed is protocol-specific, based on the protocol that the FSH object is configured to handle. For example, an HTTP Front Side Handler object can be configured to allow only HTTP 1.1 requests as well as restrict the requests to specific methods such as GET and POST.

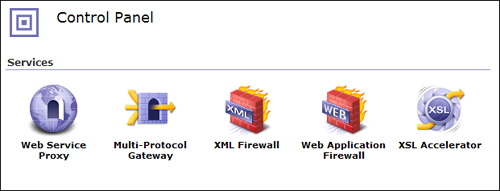
Add a note hereThere is a different FSH type that can be configured for each protocol supported on the device. For example, there is an MQ FSH, HTTP FSH, HTTPS FSH, and FTP FSHs. A single service can reference more than one FSH object enabling the service to receive requests on multiple IP address:port combinations or from more than one protocol. The valid protocol types depend on the service type, which is discussed later in this chapter.

* Add a note here**SSL Proxy—** An SSL Proxy is an object that is configured to handle the SSL communication and negotiation. This object can be referenced by a Front Side Handler object to provide the SSL connection to the client, or it can be used on the backside of the service providing the SSL connection to the backend server. The SSL Proxy determines the SSL version as well as the keys and certificates that are used in the SSL handshake. Basically, everything required to establish SSL communication is defined in this object. This object and its configuration are discussed in detail in [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)
* Add a note here**Processing Policy—** As discussed earlier in this chapter, the Processing Policy is where all the message processing takes place. It is a single object that references processing rules that apply specific processing actions to the request and response messages.
* Add a note here**Processing Rules—** A processing rule is simply an execution path that a request or response follows as it flows through the service. The processing rule references a sequence of processing actions that perform the actual processing. Separate processing rules can be configured to handle the request and the response. A processing rule can also be configured to execute in the event that an error occurs in the processing of the message.
* Add a note here**Actions—** Each execution step along the processing rule is called an action. An action can be any type of processing that can be applied to a message. There are many different types of actions that perform different functions. Some of the different types of actions include XSLT transformations, encryption, decryption, signing, logging, and AAA.
* Add a note here**XML Manager—** The XML Manager is responsible for managing many things across the entire service. There is typically only one XML Manager configured for a service. Some of the responsibilities of the XML Manager are controlling document caching, XML parser options, XSLT compile options, and much more.

## DataPower Service Types

Add a note hereAlthough every DataPower service has the same general characteristics as discussed thus far, there are different types of services that can be created on the DataPower appliance. Each type of DataPower service has built-in features and functionality to handle different types of transactions and protocols for the type of traffic that is to be expected, as well as the type of backend being proxied. The three primary services are the Web Service Proxy, Multi-Protocol Gateway, and XML Firewall.

Add a note hereFigure 6-4 shows the Services section of the Control Panel in the WebGUI, which includes these three service types. Other available service types, such as the two additional services shown in Figure 6-4, are discussed later in this chapter.

[](javascript:PopImage('IMG_88','http://images.books24x7.com/bookimages/id_30903/06fig04_alt.jpg','621','237'))  
Add a note hereFigure 6-4: DataPower services.

### Add a note hereXML Firewall

Add a note hereThe XML Firewall (XMLFW) service is designed to process generic XML requests and responses transmitted over the HTTP(S) protocol. It is conceptually one of the simplest services to understand as it can accept messages over one protocol and contains one Processing Policy with a set of request, response, two-way, and error rules providing all the multistep processing capabilities. Unlike the other main service types, the XMLFW does not contain a Front Side Handler object. The listening IP address and port are defined as a service parameter. The XMLFW also provides general threat protection against common XML threats and can be customized to provide the maximum level of threat protection while allowing valid messages to be processed.

Add a note hereAlthough the XMLFW is designed to process XML documents of all types, including SOAP-formatted messages, it can also accept unprocessed (text/binary) documents. By implementing the multistep processing via a Processing Policy, this service type can apply all the various processing actions to the request or response message regardless of the format. This can include AAA, transformations, schema validation, logging, and cryptographic operations.

Add a note hereLike all other DataPower services, the XMLFW can be configured to proxy backend services by providing a backend URL; however, one of the commonly used features of this service type is the capability to configure it as a loopback firewall. When configured as a loopback firewall, a backend server is not defined. This is because the service itself will generate a response to the client after processing the request message. This is useful when developing, testing, or debugging a service when the backend server is not available.

Add a note hereBecause a Multi-Protocol Gateway service (which is another service type discussed later in this chapter) and an XML Firewall can share the same Processing Policy, the XMLFW is often used in loopback mode for testing, allowing easy visibility into message processing without waiting for a valid service backend to be in place and responsive. This loopback capability, as well as tighter integration with its frontend HTTP(S) listener (a tradeoff introduced by its lack of flexibility for protocol mediation) makes the XML firewall a fast, efficient service for development and debugging.

Add a note hereAnother use case for a loopback firewall is to provide a standalone service. It may be possible that the DataPower service is capable of generating a response back to the client without ever calling a backend server. Because DataPower has the capability of communicating with many external resources as well as many built-in processing actions, it may be possible to fulfill the client request within the DataPower service itself. This could include a query to a database or an LDAP store based on the incoming message, or a rule that transforms the incoming message and returns the results. All this can be done within a Processing Policy returning the results back to the client without ever calling a backend server. It might be that the client is another DataPower service and the XMLFW is serving as a common “utility” service that can be called by other DataPower services. For example, suppose there is need for many of your DataPower services to perform the same series of LDAP queries to extract information about a user, encrypt some of the data returned, and format it into an XML document. This can be implemented in each DataPower service that requires it, or you can create an XMLFW in loopback mode that will perform these steps based on the user ID passed and return the formatted results."XMLFW in Loopback Mode"

**XMLFW in Loopback Mode**

Add a note hereAn XMLFW configured in loopback mode is a convenient means for testing services when the backend server is unavailable by using the firewall itself to mimic the backend server.

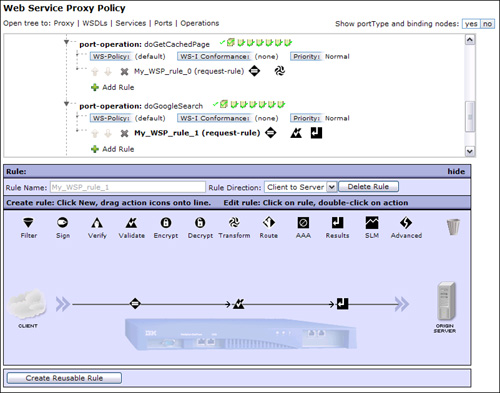
### Add a note hereWeb Service Proxy

Add a note hereA Web Service Proxy (WSP) is a DataPower service type used to provide security and abstraction for backend Web services. It is a powerful proxy for the backend Web service. By simply uploading a WSDL[[1](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=579779360&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch06fn01)] document and adding a Front Side Handler to the service, it is created and ready to start receiving requests. This may appear to be a bare bones configuration; however, it is a fully functioning, featured-filled service providing endpoint/URI abstraction, parser-based XML threat protection, XML well-formedness checking, SOAP schema validation, payload schema validation, hooks for monitoring service executions, and a platform for building operation-level rules. All these features are automatically provided as DataPower uses the information provided within the WSDL such as endpoints, schema(s) and operations to configure the service. These few simple steps provide a full-featured service to act as a base for additional configuration required to meet your use case requirements such as AAA, document transformations, message encryption, and so on.

Add a note hereIn addition to the WSP being capable of uploading or fetching a WSDL, it can also be configured to subscribe to a UDDI registry or a WSRR server to provide automatic updates of the WSDL or dynamically look up the endpoints for the service. Figure 6-5 shows the available methods for acquiring a WSDL file for use by the Web Service Proxy. As you might imagine, this flexibility can be useful for managing WSDLs and receiving updates in your service.

[](javascript:PopImage('IMG_89','http://images.books24x7.com/bookimages/id_30903/06fig05_alt.jpg','803','212'))  
Add a note hereFigure 6-5: Web Service Proxy WSDL options.

Add a note hereDataPower’s introspection of the service’s WSDL file provides a powerful opportunity to implement rules and actions that are tied closely to that service’s natural structure. For example, any of the actions available can be used in a rule and applied to a particular service operation or to every operation in the service. To demonstrate the power and flexibility of this feature, let’s take an example Web service that is used for processing online shopping requests. Suppose there is one operation in the Web service for checking out, called checkout, and another for retrieving catalog information, called browseCatalog. It may not be required to authenticate and authorize requests for the browseCatalog operation; however, the checkout operation requires authentication and authorization. Because it is possible to implement a processing rule at the operation level of the WSDL, it is easy to implement this requirement. A processing rule would simply be created to authenticate and authorize the request and applied only to the checkout operation. To illustrate this point, Figure 6-6 shows the Policy configuration screen for a Web Service Proxy. Notice the two operations doGetCachedPage and doGoogleSearch. Each of these two operations implements a different processing rule.

[](javascript:PopImage('IMG_90','http://images.books24x7.com/bookimages/id_30903/06fig06_alt.jpg','801','630'))  
Add a note hereFigure 6-6: Rules applied to the operations.

Add a note hereThe Web Service Proxy also has some powerful monitoring and logging capabilities. Web service traffic flowing through a WSP can be monitored and logged at the service level or down to the WSDL operation level, providing great flexibility and granularity in the monitoring and logging capabilities. Other features that are provided by the WSP are Service Level Monitoring (SLM) capabilities, WS-ReliableMessaging, WS-Policy, and WS-Addressing.

Add a note hereIn summary, a WSP service can be used to proxy Web services by simply importing, referencing, or subscribing to the backend WSDL file. The WSP automatically uses the endpoints contained within the WSDL to expose new, virtualized endpoints. It enforces schema validation based on the contents of the WSDL, providing an extremely powerful and valuable service with minimal configuration. The WSP can also implement all the multistep capabilities that a DataPower service has to offer, while providing many additional Web services features.

Add a note hereWhen configuring a service based on a pre-existing WSDL, you should almost *always* use a Web Service Proxy, as its automatic schema validation and port/binding/operation-level enforcement of policies and service level agreements is unmatched by any of the other service types. In an SOA infrastructure, this service type provides the deepest integration with registry, management, monitoring, and other governance tools."WSDL = WSP"

**WSDL = WSP**

Add a note hereWhen configuring a service based on a pre-existing WSDL, you should almost *always* use a Web Service Proxy as it acquires many of the configuration details from the WSDL itself.

### Add a note hereMulti-Protocol Gateway

Add a note hereThe final service of the three major service types is the Multi-Protocol Gateway (MPGW) and could be called “The Mother of All Services.” When you hear of the DataPower appliance being used as an ESB, it is typically the MPGW service that makes this possible. The MPGW is a powerful and versatile service type. In addition to all the threat protection and multistep processing capabilities of the XMLFW, the MPGW is capable of processing requests from/to various protocols. The supported protocols on the XI50 appliance include HTTP, HTTPS, WebSphere MQ, WebSphere JMS, IMS, FTP(S), NFS, and Tibco EMS.

Add a note hereSimilar to the XMLFW previously discussed, the MPGW receives incoming requests, processes them within a Processing Policy applying the configured processing actions, and forwards the request to the backend server. In fact, any Processing Policy created in an XML firewall can also be used in a Multi-Protocol Gateway. The service then processes the response similarly, applying the applicable response rule if configured. The big difference is in the multiple protocols that it can communicate with (hence, the name *Multi-Protocol* Gateway). Unlike the XMLFW, the MPGW utilizes a Front Side Handler (FSH) object to establish the connection with the client. FSHs were discussed earlier in this chapter where we mentioned that there is a specific FSH object type for each protocol supported. Leveraging the capabilities of these FSH objects, the MPGW service is capable of communicating with many different protocols.

Add a note hereBecause the MPGW service utilizes the FSH object to handle the client connection, it is also possible for the service to have multiple FSH objects listening or polling for requests. This makes it possible to receive requests using different protocols for the same service. For example, a MPGW can have one FSH listening for HTTP requests on a specified IP address and port and another FSH polling an MQ queue for messages. Both FSH objects would forward the incoming message to the service where they can be processed and passed to the backend server.

Add a note hereAll the available protocols that the MPGW can receive incoming requests over can also be used on the backend to forward the request to its destination. It is not required that the client side protocol match the backend protocol. It is perfectly valid to receive a request over one protocol such as HTTP and forward the request to a backend URL using an entirely different protocol such as FTP. This is known as protocol bridging and is a key component of an ESB.

Add a note hereBecause the MPGW service type features support for many different protocols, you might think that you would use an MPGW only when using mixed protocols or for non-HTTP(s) traffic. The truth of the matter is an MPGW would usually be used for all non-WSDL based backends even where an XMLFW would suffice. Because the XMLFW does not provide any additional functionality over an MPGW (except easily enabled loopback capabilities and perhaps speed of configuration for quick demos and tests) it would be wise to use the MPGW instead of the XMLFW. This provides the flexibility to add additional functionality in the future if needed. For example, after a service has been configured and deployed, a new requirement may dictate that the service listens on more than one IP address and port combination. With an MPGW, this can be easily accomplished by adding an additional FSH object to the already existing service.

Add a note hereIn summary, there are three major service types available in the DataPower appliance. Each of the service types provides different functionality and features for proxying different types of backend services and applications. It is important to understand these different service types and what they have to offer when planning for the creation of a new service. A general rule of thumb to guide you through the service type selection is this: If the service is proxying a Web service with an available WSDL, use a Web Service Proxy service. If there is no WSDL provided by the backend application or service, an MPGW is the next best choice. Table 6-1 shows a feature comparison of the three service types discussed.

| Add a note hereTable 6-1: Main Services Comparison Chart  [[http://www.books24x7.com/images/b24-bluearrow.gif](http://www.books24x7.com/outputobject.asp?bookid=30903&chunkid=579779360&objectid=ch06table01&objecttype=spreadsheet)Open table as spreadsheet](http://www.books24x7.com/outputobject.asp?bookid=30903&chunkid=579779360&objectid=ch06table01&objecttype=spreadsheet) | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Add a note hereService Type** | **Add a note hereLoopback Mode** | **Add a note hereMultiple protocols** | **Add a note hereMultiple listeners** | **Add a note hereWSDL Support** | **Add a note hereSLM** |
| Add a note hereWSP | Add a note hereX[[\*](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=579779360&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ftn.ch06tn01)] | Add a note hereX | Add a note hereX | Add a note hereX | Add a note hereX |
| Add a note hereXMLFW | Add a note hereX |  |  |  |  |
| Add a note hereMPGW | Add a note hereX[[\*](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=579779360&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1#ftn.ch06tn01)] | Add a note hereX | Add a note hereX |  | Add a note hereX |
| Add a note here[[\*](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=579779360&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch06tn01)]Requires additional configuration within a processing rule | | | | | |

### Add a note hereMiscellaneous Services

Add a note hereIn addition to the three main services discussed, there are additional, less frequently used service types available on the DataPower appliance. These are discussed in the following sections.

#### Web Application Firewall (WAF)

Add a note hereThe WAF service type is intended to be used to proxy Web applications, providing AAA services as well as cookie encryption, cross-site scripting protection, session timeout management, and name-value input processing/filtering. A common use of the WAF is to provide perimeter authentication for Web applications, asserting the user’s identity to the backend application server in a format that can be consumed by the application server. For example, the DataPower WAF can authenticate the user with the credentials passed in the request via HTTP basic authentication and create an LTPA token that can propagate the identity to a backend WebSphere Application Server. This functionality can also be implemented in other service types; however, the WAF provides a wizard for creating the service that will guide you through this configuration as well as the additional features for proxying Web applications mentioned previously.

Add a note hereAlthough this service type may appear to be the best choice to proxy all Web applications, there may be cases where it is not. It is important to note that the tools available in the WebGUI for configuring the WAF are different than the tools used for configuring all other services. For example, it does not provide any drag-and-drop functionality, such as the policy editor, where the actions can be dragged on the policy line and configured. The terminology used for this service type is also different from other services. For example, a rule in other services is called a map in the WAF. This is discussed in more detail in [Chapter 11](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1832#1832), [“Proxying Web Applications with DataPower.”](http://www.books24x7.com/assetviewer.aspx?bkid=30903&destid=1832#1832)

#### XSL Accelerator

Add a note hereThe XSL Accelerator validates and transforms incoming or outgoing XML documents. An XSL Accelerator service would proxy a backend service, applying all the necessary schema validation and transformations to the incoming document before the message reaches the backend service. Alternatively, it can provide content rendering of outbound messages from XML to other formats (XML, HTML, and WML). This functionality can also be provided by the XMLFW or MPGW. The primary use of this service type is on a DataPower XA35 appliance where other service types are not available.

#### XSL Coprocessor

Add a note hereLike the XML Accelerator service, the XSL Coprocessor (coproc) provides schema validation and transformations for a backend service. However, the XSL Coprocessor is a loopback service that can only be called by a Java application communicating with the service via the JAXP API. The calling Java application would pass the XSL Coprocessor service the XML document to be validated, parsed, and transformed. The service would then perform the required validation and transformation, returning the results back to the Java application. Although this is a supported service type on the appliance, it is not a recommended practice. The preferred method is to keep the DataPower service inline rather than as a coproc. This provides for much easier configuration of the service, as well as the backend application. This also allows the service to scan the incoming messages for threats before the message reaches the backend application. The XSL Coprocessor service type is utilized by the Eclipse/RAD coproc and the XML Spy plugins.

#### HTTP Service

Add a note hereAn HTTP Service can be configured to serve documents that are stored on the local DataPower appliance to HTTP clients. A service of this type would allow access to certain files on the DataPower file system without requiring a log in to the DataPower appliance. This can be thought of as a mini HTTP/Web server. This functionality is generally superseded by a robust, enterprise-class Web server deployed in your IT architecture; however, it is an available service option. The use of this service type is not a recommended practice as it is not the intent of the appliance and can deplete available memory on the device when storing many or large files on the file system.

Add a note here[[1](http://www.books24x7.com/assetviewer.aspx?bookid=30903&chunkid=579779360&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1" \l "ch06fn01)]A WSDL file provides critical information about a Web service, including endpoint locations, instructions on binding to those endpoints, and expected message schemas.

## Summary

Add a note hereIn this chapter, we explained what a DataPower service is, as well as the components and objects that make up a service. We discussed the different types of DataPower services, illustrating the value and key differentiators in each, as well as some general advice on choosing a service type. At this time you should have a good understanding about DataPower services so that you can move forward to subsequent chapters where the configuration of these services is described in depth.