

Managing IBM WebSphere DataPower Device Configurations for High Availability, Consistency, and Control

John E. Rasmussen, (rasmussj@us.ibm.com), Senior IT Specialist, IBM WebSphere DataPower

December 19, 2007

Introduction

IBM WebSphere DataPower SOA Appliances (DataPower), (DataPower devices) are purpose built network appliances that offer a wide variety of functions such as the securing and management of SOA Applications, Enterprise Service Bus Integration, and high speed XSL execution.

This is part one of a two part series. This document will describe:

- Techniques that will ensure that DataPower devices are managed in a consistent and controlled manner in a multi-device and highly available configuration
- Configuration methods and examples including WebGUI, Command Line Interface and XML Management Interface
- The migration of configuration specifications from development through production
- Potential migration issues
- The use of DataPower configuration options for migration issue mediation including XML Identity Document
- Sample configuration methods
- The use of IBM WebSphere ITCAMS SE for DataPower in configuration management
- Configuration Modification using XSLT and XML Identity Document

The second part will describe the implementation of automated processes used to manage a configuration as described in part one.

This document is not intended to describe DataPower configuration in detail, but rather configuration management techniques. For more on DataPower, please visit <http://www-306.ibm.com/software/integration/datapower/xi50/>.

Table of Contents

Managing IBM WebSphere DataPower Device Configurations for High Availability, Consistency, and Control	1
December 19, 2007	1

Introduction.....	1
1. DataPower Configuration	3
1.1. File System Directories.....	3
1.2. Application Domains	4
1.3. Devices and Environments	4
1.4. Load Balancers and Active/Active, Active/Standby Configuration	5
2. Configuration Options	5
2.1. WebGUI.....	5
2.2. Command Line Interface	6
2.3. XML Management Interface.....	7
2.4. Configuration Persistence	7
3. Configuring for Migration	8
3.1. Network objects	9
3.1.1. Host Alias.....	9
3.1.2. DNS.....	10
3.1.3. Static Hosts	11
3.2. Sample Application.....	12
3.3. XSLT Issues.....	17
3.4. Minimal Migration.....	21
3.5. Configuring for Migration Summary.....	22
4. Configuration Migration Tools	22
4.1. Package Importing and Exporting	22
4.1.1. WebGUI Methods.....	22
4.1.1.3. Importing Packages.....	25
4.1.2. CLI Methods	26
4.1.2.1. Export Functions.....	26
4.1.2.2. Import Functions.....	27
4.1.3. XML Management Methods.....	28
4.1.3.1. Export Functions.....	28
4.1.3.2. Import Functions.....	29
4.2. Package Importing and Exporting Summary	29
5. Configuration Structure for High Availability, Consistency, and Control	30
5.1. Device Configuration.....	30
5.2. Domain Configuration	31
5.3. Including Configuration.....	31
5.4. Importing Configuration	32
5.5. Including Files	32
5.6. Sample Device Configuration.....	32
5.6.1. Caution when editing device configuration file.....	33
5.8. IBM Tivoli Composite Application Manager System Edition for WebSphere DataPower.....	37
5.9. Device Configuration Summary	37
6. Configuration Promotion	37

1. DataPower Configuration

Each DataPower device contains a configuration. The device is configured via objects which are hierarchically organized into Services. It is the services which expose ports for the consumption of traffic over supporting protocols such as (s)HTTP, (s)FTP, MQ, JMS and NFS. Services implement functionality such as Authentication and Authorization of Web Services, Acceleration of XSLT Transformations, and Enterprise Service Bus Protocol Mediation.

The following is a brief discussion of DataPower architectural basics, and configuration options.

1.1. *File System Directories*

The DataPower File System is an encrypted RAM data source separated into several named 'Directories'. Directories are used to manage configuration data, store XSLT stylesheets, capture logging events, manage cryptographic certificates and keys, and control other system functions.

Configuration files are stored in the 'config:' directory, while custom data maintained by the user are stored in the 'local:' directory. The device stores most of its required files in a directory called 'store;'. You'll find a complete description of the File System in the WebGUI Guide for your device.



[Refresh Page](#)

Available Space: 88 MBytes (encrypted), 232 MBytes (temporary)













Manipulate Checked Files: <input type="button" value="Delete"/> <input type="button" value="Copy"/> <input type="button" value="Rename"/> <input type="button" value="Move"/>				
Name		Action	Size	Modified
 cert:		Actions...		
 config:		Actions...		
 export:		Actions...		
 image:		Actions...		
 local:		Actions...		
 logstore:		Actions...		
 logtemp:		Actions...		
 pubcert:		Actions...		
 sharedcert:		Actions...		
 store:		Actions...		
 tasktemplates:		Actions...		
 temporary:		Actions...		
Manipulate Checked Files: <input type="button" value="Delete"/> <input type="button" value="Copy"/> <input type="button" value="Rename"/> <input type="button" value="Move"/>				

Figure 1: DataPower File System

1.2. Application Domains

The DataPower file system is initially booted as a single data space within a ‘Domain’ or Partition labeled ‘default’. The file system may be further partitioned with the creation of Application Domains. A newly created domain will be provided with its own ‘local:’ directory, and will inherit read access to DataPower configuration rooted in the ‘store:’ directory of the default Domain. Access to other Application Domains may also be granted.

1.3. Devices and Environments

Each DataPower device may participate as a member of a peer group which provides services and shares management information with other members for the group. For the purpose of this document, a shared ‘Environment’ of devices refers to the group of devices servicing Software Life Cycle areas such as test, development, or production. Migrating configuration information to an environment refers to the entire group of devices.

1.4. Load Balancers and Active/Active, Active/Standby Configuration

It is often the case that groups of devices, perhaps an entire environment, will reside behind a Load Balancer which acts as a façade exposing a single Virtual IP Address for request traffic. Common routing algorithms such as Round Robin and Least Frequently Used are available. This topology has no effect on the configuration of the device. There will be no device affinity, as each transaction should be able to be processed by any of the devices in the Load Balanced Group. Service Level data is shared by devices via peer group registration, and all transactions participate in Service Level Monitoring.

DataPower devices may also be deployed in an Active/Active, or Active/Standby configuration whereby a Virtual IP Address is used to control multiple devices. In this configuration the standby device monitors the health of the active device, and assumes its IP Address in the event that it becomes unavailable. In an Active/Active configuration, multiple standby configurations are employed. Each device acts as the standby for the other, and assumes its traffic in the event of failure, without the need to maintain one box offline.

2. Configuration Options

There are several ways to configure the device. Those used will be dictated by requirement. They consist of Graphical Tools and Automated/Scripted and Programmatic processes. These methods may be used in conjunction, and all methods result in a similar implementation on the device.

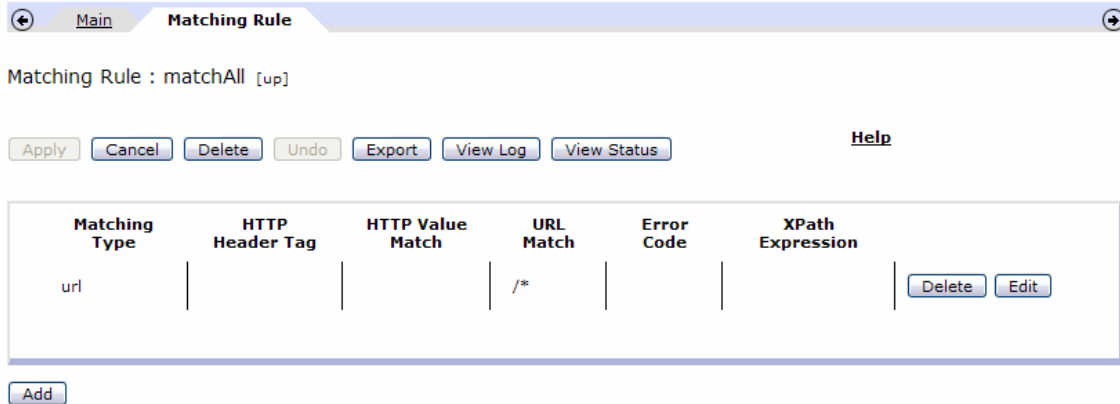
The following is a brief description of the most commonly used configuration methods.

2.1. WebGUI

Most users will configure the device via the Web-Management Graphical User Interface. This tool allows for the configuration of all of the devices' Services and supporting objects. It also provides a full range of configuration management tools. Access to the WebGUI is provided over a Secured and Authenticated HTTP Interface, and fine grained access may be defined allowing for limited access rights and privileges.

This screen shot shows the creation of a Matching Rule Object, which is used to determine the Policy to apply to transactions traversing the DataPower device.

Configure Matching Rule



Matching Type	HTTP Header Tag	HTTP Value Match	URL Match	Error Code	XPath Expression
url			/		

Figure 2: Match Rule Configured via WebGUI

2.2. Command Line Interface

Network engineers may work with devices that have been previously configured and require a more limited set of configuration. Tasks such as the addition of users to the Administration Group may be performed in an ad-hoc manor, and repetitive tasks may be scripted for automated processing. This is well suited to the Command Line Interface, or CLI. Networking Engineers will find this interface similar to those offered by other network devices. It also provides configuration management tools. This facility is provided as an Authenticated service over the Secured Shell, or SSH tunnel. Fine grained access may be defined allowing for limited access rights and privileges.

This screen shot shows the creation of a Matching Rule Object via the CLI Interface. Notice that we have 'switched' to the 'testDomain' domain.

```

192.168.1.35 - PuTTY
xi50[testDomain] (config) #
xi50[testDomain] (config) #
xi50[testDomain] (config) #
xi50[testDomain] (config) #
xi50[testDomain] (config) # match matchAny
Modify Matching Rule configuration
xi50[testDomain] (config matching matchAny) # urlmatch "/"
xi50[testDomain] (config matching matchAny) # exit
xi50[testDomain] (config) #
xi50[testDomain] (config) #
xi50[testDomain] (config) #
xi50[testDomain] (config) #
xi50[testDomain] (config) #

```

Figure 3: Match Rule Configured via CLI

2.3. XML Management Interface

Another use case involves the programmatic modification of configuration data, and may be based on external conditions and events. For example; a Brokerage Organization may define a Service Level Agreement that limits the number of trades a client may execute in a given time frame. However based on market conditions, the parameters of this SLA may change.

This demonstrates a potential use of the XML Management API. This facility allows for the authenticated real-time modification of configuration data via SOAP Messages sent over a secured HTTPS Interface. All the functions of the WebGUI and CLI are supported using the XML Management API. Each request is packaged as a SOAP Message.

Formatting of request SOAP Documents is derived via the WSDL and XSD documents available from the store: directory of the device. It also provides configuration management tools. For information regarding request formats please refer to the WebGUI Documentation for your device.

This example SOAP request shows the creation of a Matching Rule Object via the XML Management Interface. Notice the 'testDomain' attribute on the dp:request element.

```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'>
  <env:Body>
    <dp:request domain='testDomain'
      xmlns:dp='http://www.datapower.com/schemas/management'
      <dp:set-config>
        <Matching name='matchAny'
          xmlns:dp='http://www.datapower.com/schemas/management'
          xmlns:env='http://www.w3.org/2003/05/soap-envelope'>
          <MatchRules>
            <Type>url</Type>
            <Url>.*</Url>
          </MatchRules>
        </Matching>
      </dp:set-config>
    </dp:request>
  </env:Body>
</env:Envelope>
```

Figure 4: XML Management Interface Request

2.4. Configuration Persistence

Regardless of the configuration method used, the end result of each operation is a modification to the onboard device configuration. This data is expressed as an ASCII file resident on the devices' encrypted RAM File System, and contains a list of CLI Commands. If the WebGUI or XML-Management Interface was used, their execution is translated into the corresponding CLI Commands. Upon restart of the device, the designated configuration file is loaded, and Services and objects are reconstructed via its content and references to external resources. This onboard configuration file may itself be downloaded, edited and reload onto the device. Object order is significant, subordinate objects such as the Match Rule, must be defined prior to parental objects such as the XML Firewall.

The following snippet from this file demonstrates a simple XML Firewall, and supporting Style Policy Rule and Action that performs XML Threat Protection:

```
action simpleFirewall_Rule_0_Action_0
  type results
  input 'INPUT'
  output-type default
exit

rule 'simpleFirewall_Rule_0'
  action simpleFirewall_Rule_0_Action_0
exit

matching 'matchAll'
  urlmatch '/*'
exit

stylepolicy 'simpleFirewall'
  match 'matchAll' 'simpleFirewall_Rule_0'
exit

xmlfirewall 'simpleFirewall'
  local-address 0.0.0.0 2092
  summary 'an example XML Firewall Service'
  query-param-namespace 'http://www.datapower.com/param/query'
  remote-address 127.0.0.1 1001
  stylesheet-policy simpleFirewall
  response-type unprocessed
exit
```

Figure 5: Example of ASCII Configuration File

3. Configuring for Migration

Before configuring an application, it is important to understand the methods, objects and properties that will best provide for the eventual migration to another environment. While many techniques can be retrofitted into existing configurations, more complex

architectures may be avoided when these issues are addressed up front. Implementation of DNS, Static Hosts, Host Aliases, and principles such as externalizing of end points and minimal migration will be discussed.

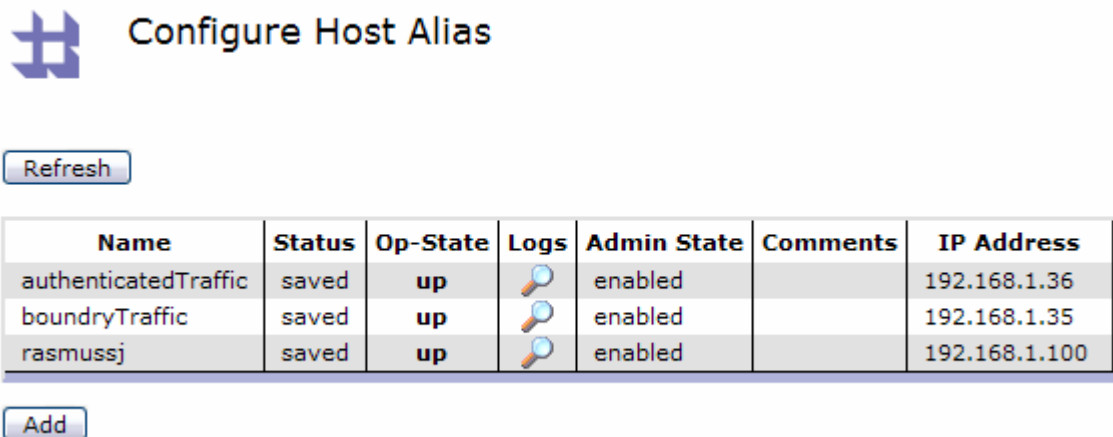
3.1. **Network objects**

The fundamental problem encountered during migration is the creation of environmental affinities within device configurations. This is often a result of the designation of off box services via dot decimal IP addresses, or the use of environment-specific DNS names. Other affinities may also be created with non IP related properties such as MQ Queue Names or Channels.

3.1.1. Host Alias

When using services such as the XML Firewall, where incoming requests may be bound to particular interfaces on the DataPower device, the use of Host Alias objects in place of numeric address of those interfaces can alleviate migration issues. The Host Alias is simply a reference to the address of the interfaces of the device.

It is recommended, that each device should have its own, unique set of Host Alias Object entries describing its interface addresses.



The image shows a web-based configuration interface titled "Configure Host Alias". It features a "Refresh" button at the top left. Below it is a table with seven columns: Name, Status, Op-State, Logs, Admin State, Comments, and IP Address. The table contains three entries: "authenticatedTraffic" with IP 192.168.1.36, "boundryTraffic" with IP 192.168.1.35, and "rasmussj" with IP 192.168.1.100. Each entry has a magnifying glass icon in the Logs column. At the bottom left is an "Add" button.





Name	Status	Op-State	Logs	Admin State	Comments	IP Address
authenticatedTraffic	saved	up		enabled		192.168.1.36
boundryTraffic	saved	up		enabled		192.168.1.35
rasmussj	saved	up		enabled		192.168.1.100

Figure 6: Host Alias


Configure XML Firewall
[Help](#)

General
Advanced
Stylesheet Params
Headers
Monitors
XML Threat Protection

Apply
Cancel
Clone

General Configuration

Firewall Name
 *


Summary

Firewall Type
 *

XML Manager
 + ... [up] *

Firewall Policy
 + ... [up] *

URL Rewrite Policy
 + ...



Back End

With a dynamic proxy back end XML Firewall type, the back end server address and port are determined by a stylesheet in a policy action.

Front End

Device Address

 *

Device Port
 *

SSL Client Crypto Profile
 + ...

SSL Server Crypto Profile
 + ... [up]

Figure 7: XML Firewall Service with Host Alias in place of Interface Address

3.1.2. DNS

The DNS Setting object provides for the designation of device specific DNS Server and Static Host settings that may be used to extend the DNS Server. As we will see, Static Hosts provide a powerful aliasing capability that will greatly assist in configuration migration. Using literal names such as; ‘MQHOST’, as apposed to dot decimal address also provides a more streamlined configuration migration. Most objects, such as the Services, Log Targets and Load Balancer Group Members can accept a literal name, (DNS or Static Host Alias), in place of dot decimal address.

If separate Test and Production DNS Server are available, use them to facilitate a streamlined migration between environments.



Configure DNS Settings

[Main](#) [Search Domains](#) **DNS Servers** [Static Hosts](#)

DNS Settings [up]

[Apply](#) [Cancel](#) [Delete](#) [Undo](#) [Export](#) [View Log](#) [View Status](#)

IP Address	UDP Port	TCP Port	Retries	
68.87.71.226	53	53	3	Delete Edit

[Add](#)

Figure 8: DNS Server Settings

3.1.3. Static Hosts

The DNS Server Object provides for the inclusion of Static Host entries where DNS Names may be assigned specific dot decimal address. This technique can be employed as a migration assistance methodology. In this case, Static Host entries in the production environment would contain different addresses than the test environment DNS Static Host entries. This technique is particularly useful when production and test environments do not share DNS names. Rather than using Domain Specific names, for example; ***test.authentication.ldap.com*** and ***prod.authentication.ldap.com***, you could use a single entry ***authentication.ldap.com*** when configuring objects, and associate the domain's specific server's address via the Static Host. When migrating application, environment-specific objects, such as the DNS Object and its Static Hosts are not migrated, thereby providing an environment neutral configuration.

It is recommended that Static Host Settings be used to disambiguate DNS names



Configure DNS Settings

[Main](#) [Search Domains](#) [DNS Servers](#) **Static Hosts**

DNS Settings [up]

[Apply](#) [Cancel](#) [Delete](#) [Undo](#) [Export](#) [View Log](#) [View Status](#)

Host Name	IP Address	
authentication.ldap.com	10.10.10.10	Delete Edit

[Add](#)

Figure 9: DNS Server with Static Host

3.2. Sample Application

In order to demonstrate fundamental configuration management and migration issues, a simple application will be used. This application consists of a Multi Protocol Gateway Service which will extract documents from a WebSphere MQ Queue, perform basic XML Threat Protection, and forward the document to a dynamically assigned backend server. The dynamic routing will be performed using an XSLT script. While the dynamic routing could have been performed using other configuration options such as xPath Routing Maps, the use of XSLT was chosen to demonstrate issues that may be confronted when using XSLT. The application will be ‘developed’ in a test environment, and migrated to a production environment.

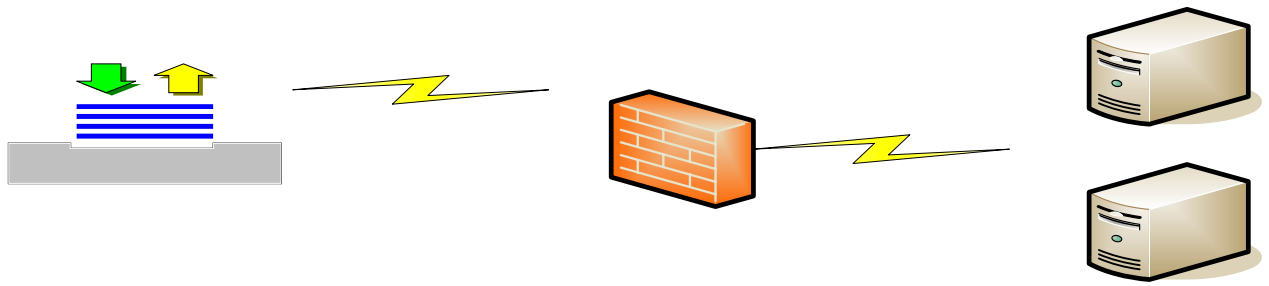



Figure 10: Sample Application

The application consists of a Multi Protocol Gateway Service, an MQ Front Side Handler, a Style Policy and it's supporting Rules, Actions, Match Rules, XML Manager, and the XSLT.

MQ Queue

3.2.1. Multi Protocol Gateway

The MPGW configuration simply designates the Front Side Handler, and assigns a Policy and dynamic backend selection.


Configure Multi-Protocol Gateway
[Help](#)

General
Advanced
Stylesheet Params
Headers
Monitors
WS-Addressing
XML Threat Protection

Apply
Cancel
Delete
Export
View Log
View Status
Show Probe

Multi-Protocol Gateway status: [up]

General Configuration

Multi-Protocol Gateway Name
 *

Summary

Type
☐ static-backend
☒ dynamic-backend

XML Manager
 + ... [up] *

Multi-Protocol Gateway Policy
 + ... [up] *

URL Rewrite Policy
 + ...

Back side settings

With a dynamic proxy back end type, the back end server address and port are determined by a stylesheet in a policy action.

Front side settings

Front Side Protocol

Type	Name	Op-State	
MQ Front Side Handler	mqFSH	[up]	Remove

User Agent settings

Match	Property
Note: To edit the User Agent, please access via the XML Manager above.	

SSL Client Crypto Profile
 + ...

Response Type
☐ SOAP
☐ XML
☒ Pass-Thru
☐ Non-XML

Request Type
☒ SOAP
☐ XML
☐ Pass-Thru
☐ Non-XML

Figure 11: MPGW Example

3.2.2. Multi-Protocol Gateway Policy

The Multi-Protocol Gateway Policy is an example of a Style Policy. While many different functions may be performed, in this case it is used to assign the XSLT used for dynamic routing selection.


Configure Multi-Protocol Gateway Policy

Select a Policy Name:
dynamicMQtoHTTP
New
Delete
View Log
View Object Status
Close

Rule Name:
dynamicMQtoHTTP_Rule_0




Create rule: Click New, drag action icons onto line. Edit rule: Click on rule, double-click on action

Entry
Rule # 1
Filter
Sign
Verify
Validate
Encrypt
Decrypt
Transform
Route
AAA
Results
Advanced
Delete



☐ Server to Client
☐ Both Directions
☒ Client to Server
☐ Error
Rule Actions:
Apply
Delete
New
Reset

Configured Rules

Reorder	Priority	Rule Name	Match Name	Direction	Actions
<div> <div> ^ v </div> </div>	1	dynamicMQtoHTTP_Rule_0	matchAll	Request Rule	  

3.2.3. MQ Front Side Handler

The Front Side Handler assigns the queue from which input documents are fetched, and the DataPower Queue Manager Object utilized. In our example the Get Queue, (the Queue from which messages are extracted), is called 'requestQueue'.



Configure MQ Front Side Handler

Main

MQ Front Side Handler : mqFSH [up]

Apply

Cancel

Delete

Undo

Export


View Log

View Status

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Queue Manager	testQueueManger (MQ Queue Manager) <input type="button" value="+"/> <input type="button" value="..."/> *
Get Queue	requestQueue *
Put Queue	<input type="text"/>
CCSI	<input type="text" value="0"/>
Get Message Options	<input type="text"/>
Exclude Message Headers	<input type="checkbox"/> CICS Bridge Header (MQCIH) <input type="checkbox"/> Dead Letter Header (MQDLH) <input type="checkbox"/> IMS Information Header (MQIIH) <input type="checkbox"/> Rules and Formatting Header (MQRFH) <input type="checkbox"/> Rules and Formatting Header (MQRFH2) <input type="checkbox"/> Work Information Header (MQWIH)

3.2.4. MQ Queue Manager

The Queue Manager Object defines, among other properties, the address and port of the off box WebSphere MQ Queue Manager. I have chosen to use a Host Alias rather than a DNS name, or dot decimal address. The use of the dot decimal addresses creates an affinity between this object and the environment. When this object is migrated, the address may no longer reference the proper MQ Queue Manager. For this reason using a hard coded address is not recommended.

 **Configure MQ Queue Manager**

This configuration has been modified, but not yet saved.

← **Main** Advanced

MQ Queue Manager : testQueueManger [up]

Apply Cancel Delete Undo Export View Log View Status

Admin State	<input checked="" type="radio"/> enabled <input type="radio"/> disabled
Comments	<input type="text"/>
Host Name	<input type="text" value="rasmussj(1414)"/> *
Queue Manager Name	<input type="text"/>
Channel Name	<input type="text" value="SYSTEM.DEF.SVRCONN"/>
User Name	<input type="text"/>
Maximum Message Size	<input type="text" value="1048576"/> bytes
Cache Timeout	<input type="text"/> seconds
Units Of Work	<input type="text" value="0"/>
Total Connection Limit	<input type="text" value="250"/>

3.3. XSLT Issues

One area that is often a cause for concern is the use of hard coded, '*Magic Numbers*' in custom XSLT code. Those readers experienced with DataPower Extension Functions may be familiar with the possible use of dot decimal address in routing statements and

other Extension Functions. The use of these hard coded addresses causes an affinity with the environment. This problem may be easily resolved by the use of the environment-specific 'Identity Document' which contains an externalized list of addresses and/or host names, (end points), used with the XSLT.

This topic will introduce the 'Identity Document'. This is a simple XML Document that can be shared by devices within an environment. It can be resident on the device or fetched from a central location such as a Source Control System. Initially we will demonstrate its utility for the externalizing and centralizing of 'aliased' configuration details. The second article in this series, will demonstrate additional facilities such as Configuration Property Modification that will leverage its implementation.

For our sample application, I've created the following Identity Document with a 'prodEnvironment' environment, containing two end points, one for two aliased hosts; MQ-Host and Finance Web Service. Multiple environments could be describe in a single Identity Document and be shared by environments, (each uniquely identified by the name attribute), or multiple documents could be used, each describing a single environment.

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- -->
<!-- j. rasmussen, rasmussj@us.ibm.com -->
<!-- This document is used to : -->
<!-- Externalize all ip/port/sslProxy information for routing
purposes -->
<!-- Remove any unwanted objects from a configuration xcfg document -
->
<!-- Modify any objects property within a configuration xcfg document
-->
<!-- -->
<identityDocument>
  <!-- -->
  <!-- An environment may contain many devices -->
  <!-- -->
  <environments>
    <environment name='prodEnvironment'>
      <!-- -->
      <!-- These addresses will be fetched from
within XSLT via xPath -->
      <!-- -->
      <endPoints>
        <MQ-Host>
          <ip>192.168.1.35</ip>
          <port>10001</port>
        </MQ-Host>
        <financeWebService>
          <ip>192.168.1.35</ip>
          <port>10002</port>
        </financeWebService>
      </endPoints>
    </environment>
  </environments>
</identityDocument>
```

Figure 12: Identity Document

In the XSLT that does the actual routing, all that needs to be done is to reference the Identities. The imported Identity Document is fetched at compilation time, and results in little or no overhead. Remember, in many instances, routing could have been performed without the use of XSLT; this example uses this technique to demonstrate the Identity Document principle.

```

<xsl:variable name='endPoints'
select='document('local:///identityDocument.xml')/identityDocument/en
vironments/environment[@name='prodEnvironment']/endPoints'/>
<!-- -->
<xsl:template match='/>
    <!-- -->
    <xsl:message dp:priority='info'>
        <xsl:value-of select='Determining Routing'/'>
    </xsl:message>
    <xsl:variable name='urlIn'
select='dp:variable('var://service/URL-in')'/'>
    <!-- -->
    <!-- Check the incoming URL, set Route -->
    <!-- -->
    <xsl:choose>
        <!-- -->
        <!-- MQ-Host routing -->
        <!-- -->
        <xsl:when test='contains($urlIn, '/MQ-Host')'>
            <dp:xset-target host='$endPoints/MQ-Host/ip'
port='$endPoints/MQ-Host/port' ssl='$endPoints/itdtest/ssl'
sslid='{ $endPoints/MQ-Host/sslid}'/'>
            <xsl:message dp:priority='info'>
                <xsl:value-of select='concat('Found
/MQ-Host/, Setting routing to ', $endPoints/MQ-Host/ip, ':',
$endPoints/MQ-Host/port)'/'>
            </xsl:message>
        </xsl:when>
        <!-- -->
        <!-- financeWebService routing -->
        <!-- -->
        <xsl:when test='contains($urlIn,
'/financeWebService')'>
            <dp:xset-target
host='$endPoints/financeWebService/ip'
port='$endPoints/financeWebService/port'
ssl='$endPoints/financeWebService/ssl'
sslid='{ $endPoints/financeWebService/sslid}'/'>
            <xsl:message dp:priority='info'>
                <xsl:value-of
select='concat('Found /financeWebService/, Setting routing to
', $endPoints/financeWebService/ip, ':',
$endPoints/financeWebService/port)'/'>
            </xsl:message>
        </xsl:when>
        <!-- -->
        <xsl:otherwise>
            <xsl:message dp:priority='error'>
                <xsl:value-of select='concat('Unknown
routing based on URL ', $urlIn)'/'>
            </xsl:message>
        </xsl:otherwise>
        <!-- -->
    </xsl:choose>
    <!-- -->
</xsl:template>

```

Figure 13: XSLT Importing Identity Document

It is recommended that ‘Magic Numbers’ never be coded in XSLT, and that a strategy such as the ‘Identity Document’ be used to externalize this domain specific data.

3.4. Minimal Migration

Certain objects such as Ethernet Interfaces and DNS Settings are environment-specific. They should not be part of any on going migration of applications. Once an application has been developed, when and if changes are made, only those objects which have been modified should be migrated. The majority of the time, this will include objects such as Style Policy Rules, Actions, AAA Policies, and not these environment-specific objects. This strategy will greatly reduce the need to handle these potential migration issues.

For example an Application Domain Export of our test domain contains the following objects:

Object Type	Object Name
LogTarget	default-log
Statistics	default
HTTPUserAgent	default
ProcessingMetadata	ftp-usercert-metadata
ProcessingMetadata	ftp-username-metadata
SLMAction	notify
SLMAction	shape
SLMAction	throttle
StylePolicyAction	dynamicFirewall_Rule_0_Action_0
StylePolicyAction	dynamicFirewall_Rule_0_Action_1
StylePolicyAction	dynamicMQtoHTTP_Rule_0_Action_0
StylePolicyAction	dynamicMQtoHTTP_Rule_0_Action_1
StylePolicyRule	dynamicFirewall_Rule_0
StylePolicyRule	dynamicMQtoHTTP_Rule_0
Matching	matchAll
Matching	matchAny
StylePolicy	default
StylePolicy	dynamicFirewall
StylePolicy	dynamicMQtoHTTP
XMLManager	default
WSStylePolicy	default

MQQM	testQueueManger
MQSourceProtocolHandler	mqFSH
MultiProtocolGateway	dynamicMQtoHTTP
WebServicesAgent	default
NFSDynamicMounts	default
ImportPackage	testDomainMPGW

Table 1: Object List from domain export

In all likelihood only the objects related to application function such as the MQ Queue Manager, Multi Protocol Gateway, MQ Source Protocol Handler and the Policy objects such as Matching Rules, Rules and Actions need be migrated. By selectively choosing these objects when creating the export, ancillary objects may be left out of the exported configuration.

It is recommended that only objects that require migration be included in configurations imported or included into other devices/environments.

3.5. Configuring for Migration Summary

We have seen how the use of configuration options such as Host Alias, Static Hosts, environment-specific DNS Servers and externalizing ‘Magic Numbers’ in XSLT can be very effective in easing the issues in configuration migration. Some objects will have to be hand edited when initially establishing a device. For example, each device will have to have specific Ethernet Interface definitions. However once a device is established, the need for maintaining these objects will be minimal.

4. Configuration Migration Tools

4.1. Package Importing and Exporting

There are several methods for the exporting and importing of configuration details. All three of the primary configuration methods; WebGUI, CLI and XML-Management Interface may be used. In addition configuration modification may be performed by editing the configuration file on the device. As we demonstrated earlier the on board configuration file located in the config: directory contains a sequential list of CLI commands. The methods may very well be used in conjunction; you will see that an export created by the WebGUI may be referenced by an invocation of the CLI or the XML-Management Interface for example.

4.1.1. WebGUI Methods

4.1.1.1. Export Functions

The simplest way to move configuration between environments is to use the Import/Export Utilities of the WebGUI. This tool allows for the export of configuration specifications, and allows for the convenient automatic fetching of all ‘referenced’ objects. For example in our sample application, the Multi Protocol Gateway referenced Front Side Handlers, MQ Queue Manager Object, Match Rules, Style Policy Rules and Actions, and other objects. This is a simple way to fetch all these objects in a single action, but as we will see, it may not always be the best method to use. Rather than simply migrating all reference objects it will be more practical to only migrate the objects that have been modified. If a change is made to the routing Style Policy Rule, only that rule need be migrated, not all the reference objects such a Match Rule, Service Object, etc.

The following screen shot shows the exportation of the Multi Protocol Gateway, and its referenced objects.

Figure 14: Export Utility

4.1.1.2. Export Format

The configuration may be exported as a ZIP or XML file. In either case, the details of the configuration are contained as an XML Document. When objects are exported via the WebGUI or XML-Management Interface, you may select to export the subordinate

objects referenced. The following table lists the objects selected when exporting the Multi Protocol Gateway of our sample application:

Object Type	Object Name
MQQM	testQueueManger
MQSourceProtocolHandler	mqFSH
HTTPUserAgent	default
XMLManager	default
Matching	matchAll
StylePolicyAction	dynamicMQtoHTTP_Rule_0_Action_0
StylePolicyAction	dynamicMQtoHTTP_Rule_0_Action_1
StylePolicyRule	dynamicMQtoHTTP_Rule_0
StylePolicy	dynamicMQtoHTTP
MultiProtocolGateway	dynamicMQtoHTTP

Table 2: Object exported with MPGW

The format of the XML Export document is easy to decipher. Note that the XSLT files referenced by the Service are also exported. They are encoded in base64 format.

```

<datapower-configuration version="3">
  <export-details>
  <interface-data>
  <configuration domain="testDomain">
    <MQQM name="testQueueManger" xmlns:dp="http://www.datapower.com/schemas/management" xmlns:env="
http://www.w3.org/2003/05/soap-envelope">
    <MQSourceProtocolHandler name="mqFSH" xmlns:dp="http://www.datapower.com/schemas/management" xmlns:env="
http://www.w3.org/2003/05/soap-envelope">
    <HTTPUserAgent name="default" intrinsic="true" xmlns:dp="http://www.datapower.com/schemas/management"
xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <XMLManager name="default" intrinsic="true" xmlns:dp="http://www.datapower.com/schemas/management"
xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <Matching name="matchAll" xmlns:dp="http://www.datapower.com/schemas/management" xmlns:env="
http://www.w3.org/2003/05/soap-envelope">
    <StylePolicyAction name="dynamicMQtoHTTP_Rule_0_Action_0" xmlns:dp="
http://www.datapower.com/schemas/management" xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <StylePolicyAction name="dynamicMQtoHTTP_Rule_0_Action_1" xmlns:dp="
http://www.datapower.com/schemas/management" xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <StylePolicyRule name="dynamicMQtoHTTP_Rule_0" xmlns:dp="http://www.datapower.com/schemas/management"
xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <StylePolicy name="dynamicMQtoHTTP" xmlns:dp="http://www.datapower.com/schemas/management" xmlns:env="
http://www.w3.org/2003/05/soap-envelope">
    <MultiProtocolGateway name="dynamicMQtoHTTP" xmlns:dp="http://www.datapower.com/schemas/management"
xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  </configuration>
  <files>
    <file name="local:///dynamicRoute.xsl" src="local/dynamicRoute.xsl" location="local" hash="

```


Figure 15: Exported Configuration Details

4.1.1.3. Importing Packages

Importing packages via the WebGUI is a simple and straightforward process. Simply use the Import facility, and select the exported ZIP or XML bundle. You can selectively import individual objects from the list if you desire, and you will be alerted if the objects are already on the target platform, and if they are identical to the objects on the target platform.



Import Configuration

Details of Configuration to Import

Domain Name:	testDomain
Comment:	
User Name:	admin
Device Name:	DataPower XI50
Firmware Version:	XI50.3.6.0.20
Export Timestamp:	2007-08-30 11:04:22 EDT

Contents of Configuration to Import

The following configuration objects already exist:

Check to Overwrite:

- ☐ MQ Queue Manager **testQueueManger**
- ☐ MQ Front Side Handler **mqFSH**
- ☐ User Agent **default**
- ☐ XML Manager **default**
- ☐ Matching Rule **matchAll**
- ☒ Processing Action **dynamicMQtoHTTP_Rule_0_Action_0**
- ☒ Processing Action **dynamicMQtoHTTP_Rule_0_Action_1**
- ☒ Processing Rule **dynamicMQtoHTTP_Rule_0**
- ☒ Processing Policy **dynamicMQtoHTTP**
- ☐ Multi-Protocol Gateway **dynamicMQtoHTTP**

The following files are identical to existing files:

- ☐ **local:///dynamicRoute.xsl**

Figure 16: WebGUI Import

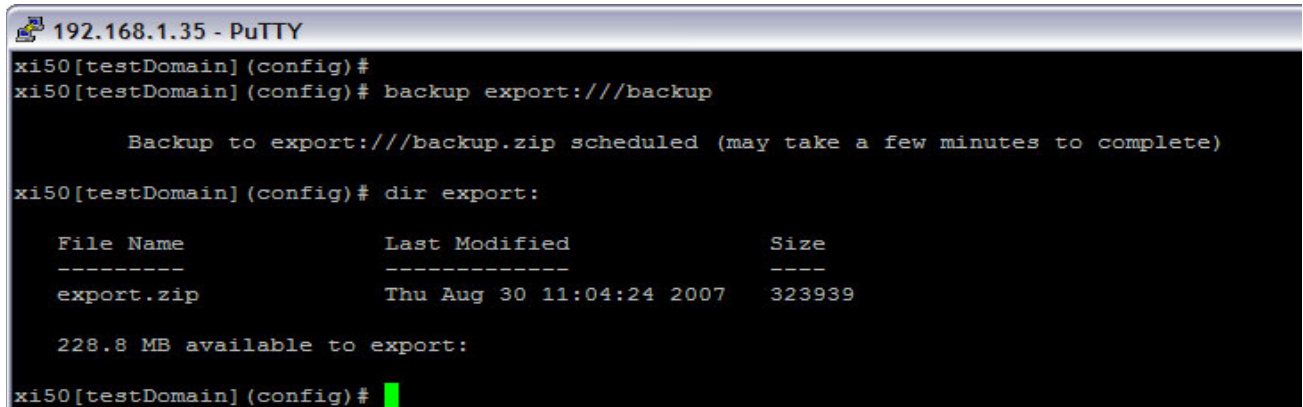
4.1.2. CLI Methods

The CLI has the advantage of providing for an automated, scripted execution of Import/Export functions.

4.1.2.1. Export Functions

The export functions of the CLI are performed via the 'Backup' command. This functionality is equivalent to the 'backup entire domain' capability of the WebGUI, and does not support the selective filtering of objects.

It is important to note that the CLI does not support selective export of services or objects. However the CLI can be used to import a selective export created by the WebGUI or XML-Management Interface.



```
192.168.1.35 - PuTTY
xi50[testDomain] (config)#
xi50[testDomain] (config)# backup export:///backup

Backup to export:///backup.zip scheduled (may take a few minutes to complete)

xi50[testDomain] (config)# dir export:

File Name          Last Modified      Size
-----
export.zip          Thu Aug 30 11:04:24 2007  323939

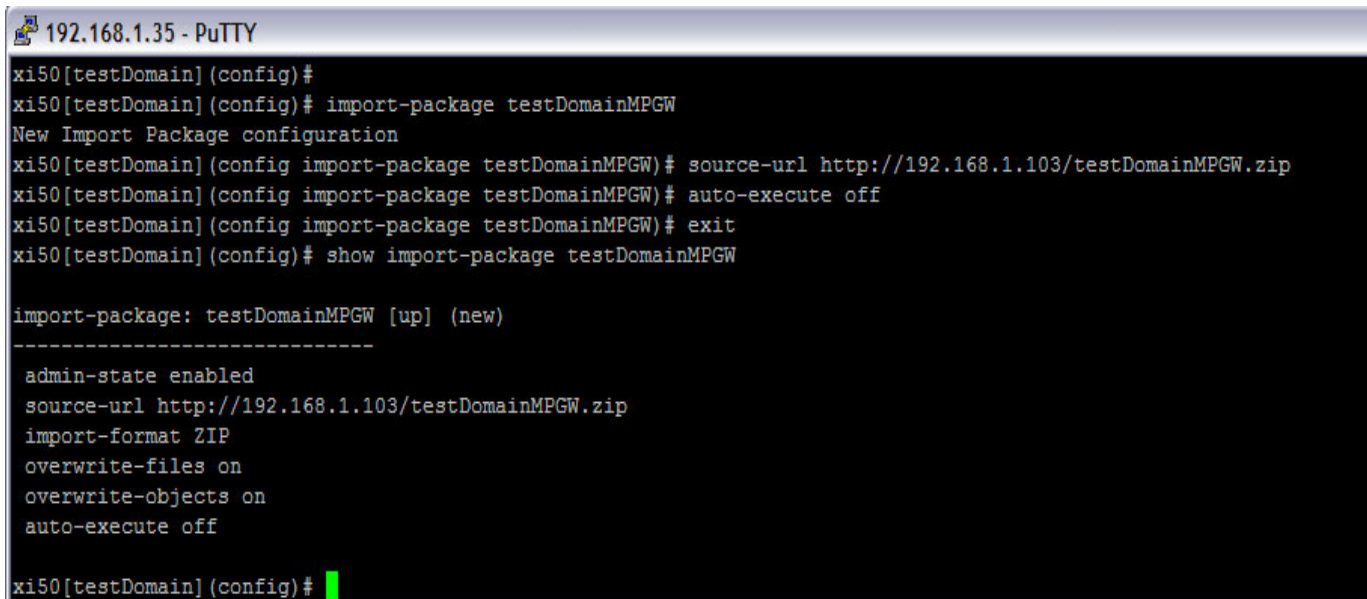
228.8 MB available to export:

xi50[testDomain] (config)#
```

Figure 17: Backup of configuration via CLI

4.1.2.2. Import Functions

The CLI supports the Importing of packages created by all Export methods. This process is performed by first creating an 'Import Package', and then executing the package. The Imported Package could have been created via another tool such as the WebGUI, or XML Management Interface. Several options such as turning on/off overwriting of files and objects are supported, as is the ability to automatically perform the import at device startup. Notice in this instance the auto-execute property has been turned off, thereby avoiding the automatic execution of this import at startup.



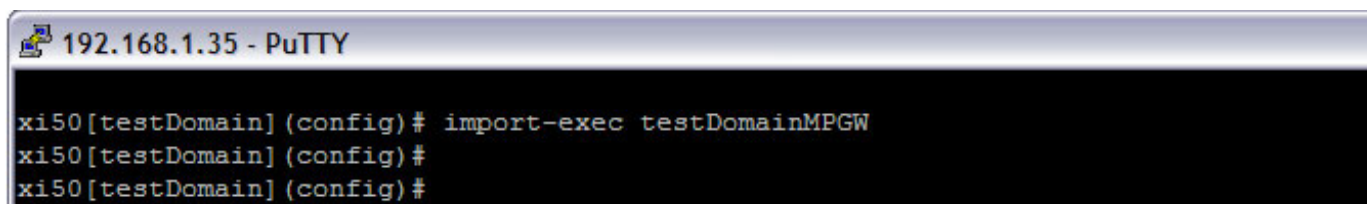
```
192.168.1.35 - PuTTY
xi50[testDomain] (config)#
xi50[testDomain] (config)# import-package testDomainMPGW
New Import Package configuration
xi50[testDomain] (config import-package testDomainMPGW)# source-url http://192.168.1.103/testDomainMPGW.zip
xi50[testDomain] (config import-package testDomainMPGW)# auto-execute off
xi50[testDomain] (config import-package testDomainMPGW)# exit
xi50[testDomain] (config)# show import-package testDomainMPGW

import-package: testDomainMPGW [up] (new)
-----
admin-state enabled
source-url http://192.168.1.103/testDomainMPGW.zip
import-format ZIP
overwrite-files on
overwrite-objects on
auto-execute off

xi50[testDomain] (config)#
```

Figure 18: Creation of Import-Package

The import function can be performed by the execution of the ‘import-exec’ command. Note that is an asynchronous function, and no feedback is provided.



```
192.168.1.35 - PuTTY

xi50[testDomain] (config)# import-exec testDomainMPGW
xi50[testDomain] (config)#
xi50[testDomain] (config)#
```

Figure 19: Import of Package via CLI

4.1.3. XML Management Methods

The XML Management Interface provides a rich variety of capabilities similar to the WebGUI. In regard to Import/Export functionality; specific objects, classes of objects, or entire domains may be exported to XML or ZIP formats.

4.1.3.1. Export Functions

The following document demonstrates a request to export the individual Multi Protocol Gateway Service that was exported via the WebGUI. The file exported will be returned in a dp:response document node containing the base64 encoded version of the exported configuration in the format specified.

```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'>
  <env:Body>
    <dp:request
xmlns:dp='http://www.datapower.com/schemas/management'>
      <dp:do-export format='ZIP' all-files='false'>
        <dp:user-comment>This is a test of the SOMA Export
extension.</dp:user-comment>
        <dp:object class='MultiProtocolGateway'
name='dynamicMQtoHTTP' ref-objects='true' ref-files='true' />
      </dp:do-export>
    </dp:request>
  </env:Body>
</env:Envelope>
```

Figure 20: Object Export via XML Management Interface

4.1.3.2. Import Functions

The XML Management Interface also allows for the programmatic importation of configuration. The do-import request can be used to import a previously exported configuration. The reference can be a file on the device, or another location off the device using HTTP(S), (S)FTP, (S)CP, (S)FTP, or other supported protocols. Files may be referenced by URL, or packaged inline in base64 format.

```
<?xml version='1.0' encoding='UTF-8'?>
<env:Envelope xmlns:env='http://schemas.xmlsoap.org/soap/envelope/'>
  <env:Body>
    <dp:request domain='testDomain'
xmlns:dp='http://www.datapower.com/schemas/management'>
      <dp:do-import source-type='XML' overwrite-
files='true' overwrite-objects='true'>
        <dp:input-
file>http://192.168.1.103/export.xcfg</dp:input-file>
      </dp:do-import>
    </dp:request>
  </env:Body>
</env:Envelope>
```

4.2. *Package Importing and Exporting Summary*

The DataPower device supports a wide variety of methods for the exporting and importing of configuration details. While the WebGUI offers the simplest methods, the XML Management Interface, and the CLI provide the backbone for an automated, scripted device configuration management capability.

We have also discussed methods that may be employed to minimize the need to modify configuration during migration. The following Best Practices will provide for a cleaner configuration and one that eliminates, or at least minimizes migration issues:

- *Use Host Alias rather than dot decimal address in Services that expose external ports*
- *Use Environment Specific DNS when possible rather than dot decimal address in configuration*
- *Use Static Hosts to handle DNS aberrations*
- *Externalized XSLT IP/Port and Host Name references via the Identity Document*
- *Migrate only those objects which require migration*

5. Configuration Structure for High Availability, Consistency, and Control

Now that we have discussed configuration best practices and methods for the exportation and importing of configuration details, we can demonstrate methods for their utilization in DataPower device configuration. The objectives are as follows:

- Provide for a consistent configuration across devices within an environment
- Provide for a consistent configuration though device restart
- Utilize Source Control for configurations
- Segregate environment-specific from shared default domain and Application configuration

5.1. Device Configuration

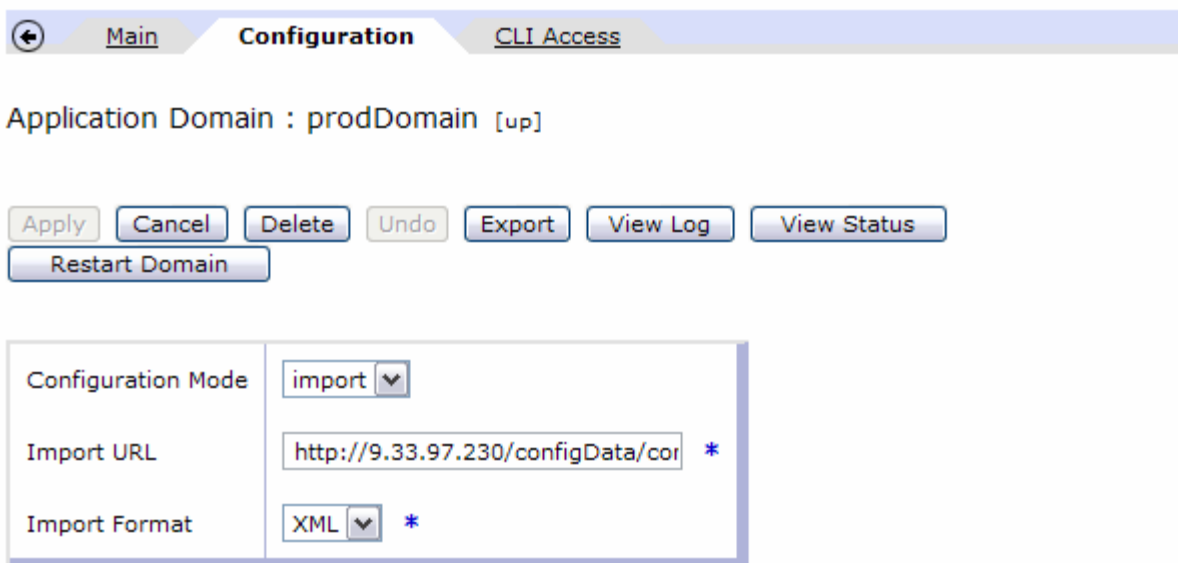
Each device will contain a section of its configuration that is unique from all other devices. At a minimum this will be the definition of its Ethernet Interfaces, but may contain other objects such as System Contact information. No other configuration can be fetched from external resources without first configuring the interfaces.

Additionally, each device will contain a section of its configuration that represents the details of ‘applications’ that it supports. Applications consist of the Services and their supporting Style Policies, Rules, Actions, AAA Policies, Match Rules, and XSLT, among other objects. In general applications will be supported among all devices in an environment. Rather than having this application level of configuration reside on the device, it is desirous that it be fetched from an external source, and furthermore that this resource be under the control of a Source Control System. In the following examples, techniques that may be employed to achieve this goal will be demonstrated.

5.2. Domain Configuration

Each domain may be configured to ‘fetch’ the shared components of its configuration from a local directory, or external source at device startup. By using this technique, the device will refresh itself to a ‘known good’ configuration at each controlled restart, or even in the event of an uncontrolled restart should for example a power interruption be encountered. This example illustrates the ability to fetch Application Domain configuration detail, previously exported from another operation, from a Source Control System.

Configure Application Domain



← Main Configuration CLI Access

Application Domain : prodDomain [up]

Apply Cancel Delete Undo Export View Log View Status

Restart Domain

Configuration Mode	import ▼
Import URL	http://9.33.97.230/configData/cor *
Import Format	XML ▼ *

Figure 21: Domain Configuration

5.3. Including Configuration

Another method of configuration fetching is provided by the ‘Exec’ command, which may be executed from the CLI, or included into the configuration file of the device. While the following example uses HTTP, (S)CP, (S)FTP, NFT or any other supported protocols may also be used. The configuration may have been fetched from a Source Control System, in this example however the config: directory is used to demonstrate access from the devices file system.

```
top;
configure terminal;

matching 'matchAny'
  urlmatch '.*'
  match-with-pcre
exit;
```

Figure 22: config:///matchAny.cfg

```
xi50[testDomain] (config)# exec config:///matchAny.cfg
Global configuration mode
Modify Matching Rule configuration
xi50[testDomain] (config)#
```

Figure 23: Exec of external config

5.4. Importing Configuration

As previously demonstrated the “exec-import” command can be used from the CLI or included into the configuration file of the device to perform an Import of a previously Exported configuration package. In addition, the package itself can be made to automatically import at startup with the use of the ‘auto execute’ property of the import-package.

5.5. Including Files

While the previous discussion has dealt with configuration packages, often individual files; XSLT, or XML may need to be imported into the device at startup. This is easily performed with the ‘Copy’ command. Again all supported protocols may be used. This example illustrates the ability to fetch configuration detail from a Source Control System.

```
copy -f http://9.33.97.230/configData/configFiles/setErrorCode.xml
local:setErrorCode.xml;
```

5.6. Sample Device Configuration

As we have demonstrated, each device will have a ‘fixed’ configuration component that contains, at a minimum, the definition of the devices’ Ethernet Interfaces. Many devices in an environment will share default domain configuration of objects such as DNS and SNMP. Finally in most cases, each device in an environment will share the applications and the domains they are defined within.

We can describe the configuration of our production environment as consisting of these three sections;

- Defining an onboard configuration file that contains the devices' Ethernet Interfaces
- Execution of shared configuration via 'exec', (from Source Control), and execute the remaining default domain objects including Application Domain definitions
- Application Packages from Source Control

5.6.1. Caution when editing device configuration file

Care should be exercised when directly editing the onboard device configuration file. Specifically, you want to make sure you maintain the Ethernet Interfaces. If you do not configure the interfaces, you will have no TCP access to the device. Similarly, make sure you define the WebGUI, SSH, XML-Management Interface and other configuration methods you intend to use. You can always get the current configuration by viewing or downloading the autoexec.cfg file from the config: directory of the default domain. If you do not define the Ethernet Interfaces, you can still get access to the device via the Serial Port. Please refer to the WebGUI Guide for your device for additional information.

Before you maintain the device configuration file you should:

- Make an entire device back up via the WebGUI Export Utility
- Make a back up of the default domain and all application domains
- Have a serial cable and physical access to the device as a fail-safe back up plan for device access

5.7. On Board config::///autoconfig.cfg, unique to each device:

The following example demonstrates a sample 'barebones' default domain configuration file which would be uploaded to the config: directory of the device and identified as the 'boot config' on the System Control Panel. It defines the Ethernet Interface, Host Aliases, System Contact Info, the WebGUI and the XML-Management Interface. Finally the configuration executes an off box configuration that contains configuration information shared by all devices in the environment.

```

configure terminal

# This is the fixed component of the 'Default' Domain for DataPower
device 192.168.0.52

interface "eth0"
  ip address 192.168.1.50/24
  mtu 1500
  ip default-gateway 192.168.0.1
  arp
  mode 1000baseTx-FD
exit

interface "eth1"
  ip address 192.168.1.51/24
  mtu 1500
  ip default-gateway 192.168.0.1
  arp
  mode 1000baseTx-FD
exit

system
  contact "rasmussj@us.ibm.com"
  name "prodFirewallAlpha"
  location "4th Floor Networking"
exit

host-alias "authenticatedTraffic"
  reset
  ip-address 192.168.1.50
exit

host-alias "boundryTraffic"
  reset
  ip-address 192.168.1.51
exit

# Web and XML-MGMT Interfaces can be restricted to a INT, so put them
in Fixed Config

web-mgmt
  admin-state enabled
  local-address 0.0.0.0 443
  idle-timeout 6000
exit

xml-mgmt
  admin-state enabled
  local-address 0.0.0.0 9999
  mode any+soma+v2004+amp+slm
exit

# Now pull in the Variable part of the configuration, this is the
same for all devices in this environment, i.e. Dev/Test/Prod

exec
http://9.33.97.230/configData/configFiles/prodFirewallAlpha/defaultDo
mainVariable.cfg

```

5.7.1. ‘Shared default domain’, including Application Domain definition

The following example demonstrates a sample variable component of the default domain. It configures objects such as DNS, SNMP, and NTP that are shared by all devices in the environment. It also contains the definition of the prodDomain Application domain that itself identifies an off box location for its configuration.

```

# This is the variable component of the 'Default' Domain for all
DataPower devices
# It is "exec"ed into the device via the defaultDomainFixed.cfg
configuration

dns
  admin-state enabled
  search-domain "some.com"
  name-server 152.155.21.10 53 53 0 3
  name-server 152.155.21.60 53 53 0 3
exit

alias "reload" "flash;boot config autoconfig.cfg;shutdown reload"

network
  admin-state enabled
  icmp-disable Timestamp-Reply
exit

ntp-service
  admin-state disabled
  remote-server 152.159.20.10
  remote-server 152.159.20.60
exit

timezone EST5EDT

snmp
  admin-state enabled
  ip-address 192.168.0.52
  community "public" "default" "read-only" "0.0.0.0/0"
exit

ssh 0.0.0.0 22

save-config overwrite

domain "prodDomain"
  reset
  base-dir local:
  base-dir prodDomain:
  config-file prodDomain.cfg
  visible-domain default
  url-permissions http+https+snmp+ftp+mailto+mq
  file-permissions CopyFrom+CopyTo+Delete+Display+Exec+Subdir
  config-mode import
  import-url
"http://9.33.97.230/configData/configFiles/prodDomain.xcfg"
  import-format XML
exit

```

Figure 25: Shared default domain Configuration

5.8. IBM Tivoli Composite Application Manager System Edition for WebSphere DataPower

ITCAM SE for DataPower can be used to assist in the migration of Configurations across devices in a shared environment. Using this tool, a configuration installed on a 'Master' device can be deployed across all dependent devices in an automated fashion. Therefore only the Mater Device need be maintained. The configuration management techniques described in this document still apply when using ITCAMSE for DataPower.

For more information on Tivoli Monitoring, please see <http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliMonitoring.html>.

5.9. Device Configuration Summary

Each device has the need to define unique objects such as Network Interfaces. Several objects such as DNS or NTP Servers can be shared across devices in an environment. Configurations can be fetched from external resources, and Application Domains can be defined to import their configuration at start up.

Using these techniques, there will be a minimum of duplication and potential miss-configuration along with the ability to restart devices at a known good configuration state and with a more finite set of configuration details to manage. Integration with Source Control Systems provides for an auditable and secured repository for configuration details.

6. Configuration Promotion

6.1. Migration from Development to Testing/QA

As each developer completes the unit testing of new features or modification to the DataPower configuration, changes need to be migrated to a testing platform. Only those objects that have been changed should be migrated to the testing platform. The use of minimal migration will avoid the potential corruption of configuration outside of the developers intended scope. For example if a change is made to the XSLT used in a Style Policy Rule, only the XSLT needs to be migrated to the test environment, not the Style Policy Rule. If an additional Action is added to a Style Policy Rule, then the entire Policy and its associated Rules and Actions should be migrated, but there is no need to migrate the Service itself.

6.2. Migration from Test/QA to Staging

Testing within the QA environment requires testing of complete application configurations, and is a preparation for the migration of entire domain configurations into the Source Control System. As has been previously demonstrated, application domains should fetch their configurations from Source Control Systems. Therefore the migration from Test/QA to Staging is a validation of the packaging and importation of these application domain configurations.

Using our example application domain, the follow process would be used for the migration of configuration changes through to production:

- Developer migrates changes to Test/QA environment
- Changes are validated
- Export of application domain, (as zip or xml), is made from Test/QA and staged in Source Control System
- Staging environment, which is a replication of the Production environment, fetches the new application package
- Staging environment is validated
- Application export is moved to Production Source Control
- Production environment can now use new application export

7. Summary

The emphasis of this document has been to describe the configuration options available within the DataPower architecture to provide for a manageable configuration which can survive unexpected system events and to work with Source Control Systems to provide for control over configuration resources. The use of the Identity Document has been demonstrated as a meaningful method to externalize 'Magic Numbers' from within XSLT, as well as a basis for configuration mediation and control.