WebSphere DataPower

IBM

Version 6.0.2

Secure Deployment Guide

IBM WebSphere DataPower

Revision History

13 April 2015 – Revision 1
8 October 2015 – Revision 2

Table of Contents

1 Introduction

This document provides guidance to administrators and users who wish to use IBM WebSphere DataPower Service Gateway Firmware version 6.0.2.0, IBM WebSphere DataPower XI52 Firmware Version 6.0.2.0, and IBM WebSphere B2B Appliance XB62 Firmware Version 6.0.20 (hereafter refers to as WebSphere DataPower Firmware version 6.0.2.0 or the TOE) in a certified, Common Criteria compliant, secure configuration. This document is intended to address the most important issues at a high level, and refers to existing documentation where more details are needed. However, please note that this document supersedes all other existing documentation.

WebSphere DataPower Firmware version 6.0.2.0 is the subject of this document as the Target of Evaluation (TOE) for Common Criteria certification. WebSphere DataPower Firmware version 6.0.2.0 has been evaluated under Common Criteria version 3.1 at level of assurance EAL4.

A system can be considered to be CC compliant if it matches an evaluated and certified configuration. This implies requirements pertaining to hardware, firmware, and software, as well as requirements pertaining to operating environment, users, and the ongoing operating procedures.

In part, this Secure Deployment Guide will guide you through the following:

- Required initial steps of administrative setup of the product and creation of one or more optional application domains. Creation of at least one application domain is highly recommended.
- Setup of one or more services, which accept message traffic and forward it to a destination. At least one service is required to pass message traffic through the appliance.
- Daily tasks you must do to maintain the appliance in good working order
- "As needed" tasks

This document assumes that you or someone else your organization has brought the appliance into a running state by following the instructions in the *Secure Installation Guide.*

Before we launch into a discussion of administration, we need to cover some background material.

1.1 Background

The DataPower products are special-purpose appliances that provides the following services:

- Message-level firewall services
- Protocol bridging, such as HTTP
- Message routing
- Message content transformation
- Cryptographic services (such as digital signing and verifying, and encryption and decryption of messages)
- Authentication and Authorization

Additional capabilities are also available. The DataPower products allow administrators to set firewall or gateway policies based on the following criteria:

- Presumed address of the source subject
- Presumed address of the destination subject
- Transport layer protocol
- Interface on which traffic arrives and departs
- Service (expressed as a URL)
- Contents of a message
- Identity of sender

Administrators have no direct access to the operating system or the file system, and there are no "users"[1]. In other words, the product provides no general purpose computing or storage facilities.

This Guide is a companion to the product's administrative reference manuals. You can reference the Command Line Interface (CLI) documentation contained in IBM Knowledge Center delivered with this Guide for more complete information about the CLI commands mentioned in this Guide. This Guide will discuss a "secure configuration", recommending specific choices to aid in a secure deployment and ongoing operation of the product.

Note that there are many functions in the products, not all of which have been certified for Common Criteria-evaluated use. The subset of capabilities offered by the product that have been certified for Common Criteria are herein referred to as the TOE (Target Of

[1] While the word "user" appears in this document and other Guidance documents, it always refers to an administrator.

Evaluation). **Use of functionality that is not part of the TOE will put the TOE out of its evaluated configuration.**

See the section entitled Conforming to the Evaluated Configuration for more information. For a list of capabilities and CLI commands included in the TOE, see Appendix B.

This guide will discuss some of the commands and modes of operation that are included in the TOE.

This Guide will discuss administrator choices expressed via the CLI interface. The CLI is the administrative tool that is available in direct serial connect with the TOE (that is, co-location), and hence is always available (assuming that authorized personnel are still able to physically access the TOE). Administrators can also access the CLI over the network through an SSH connection. Note that this connection capability must be activated using a serial line connection. Establishing SSH connectivity is discussed in Section 3.6.4 of this document.

Note that a large number of CLI commands are "configuration modes". The initial command puts you "into" the configuration mode. While in the mode, you can issue subcommands specific to that mode. For example, the configuration mode for creation of new administrators has a "password" subcommand.

1.2 Requirements for the Evaluated Configurations

The TOE combined with its underlying operating system and hardware is a network appliance that provides application-level firewall functionality, web service proxy functionality, and message content transformation functionality.

There are three variations of the TOE that have been Common Criteria certified. Each TOE variation provides similar functionality and corresponds to the IBM products described below.

1. The DataPower Service Gateway XG45 is a lightweight, level entry network shipped in a 1U rack system that provides:
   - service proxy
   - application-level firewall based on information flow control policy based on protocol information, message content and identity assertion (authentication)
   - message content transformation based on Xpath and XSLT
2. The DataPower Integration Appliance XI52 offers all the functionality provided by the XG45 but in a more powerful 2U rack system. In addition, it provides

support for more message formats and more connectivity options (not included in the evaluated configuration).

3. The DataPower B2B Appliance XB62 runs in a 2U rack system and provides B2B functionality in addition to the features included in the XG45 and XI52 models. It supports B2B messaging protocols such as AS1, AS2, AS3, and ebMS (not included in the evaluated configuration).

1.2.1 Hardware Requirements

The hardware upon which the TOE executes is part of the Operational Environment. Each of the TOEs that has been evaluated requires the following network appliance hardware models:

- IBM WebSphere DataPower Gateway XG45 Firmware: Type 7198
- IBM WebSphere DataPower Integration Appliance XI52 Firmware: Type 7199
- IBM WebSphere DataPower B2B Appliance XB62 Firmware: Type 7199

1.2.2 Operating System Requirements:

The operating system upon which the TOE executes is part of the Operational Environment. The operating system for the TOE is packaged with the TOE as part of the firmware package and is required by the evaluated configuration.

1.2.3 Other Requirements:

In addition, the operational environment may include one or more systems against which the TOE connect to enforce certain authentication and authorization rules defined in the information flow policies. Such authentication and authorization systems include:

- LDAP server
- ClearTrust server
- Netegrity SieMinder server
- RADIUS server
- SAML responder
- SAML server
- Tivoli Access Manager
- WB-Trust server

1.2.4 Physical Requirements

The TOE must be protected against unauthorized physical access and modification.

Systems providing authentication and authorization services to the TOE must be protected against unauthorized physical access and modification in the operational environment.

Network services running in the operational environment that are used by the TOE must be reliable and protected against unauthorized physical access and modification

1.2.5 Personnel Requirements

The administrators of the TOE and the systems in the TOE"s operational environment involved in protecting TSF data or providing functionality that the TOE depends on, must not be careless, willfully negligent, or hostile. They must follow and abide by the instructions provided in the administrator guidance that is part of the TOE. They are well trained to securely and responsibly administer all aspects of the TOE operations in accordance to the Security Target.

The administrators of the TOE must make sure that digital certificates imported into the TOE and CRLs used for certificate validation must meet X.509 standard requirements. In addition, digital certificates must be generated with a key length and using a message digest algorithm that provide sufficient security strength.

Please refer to section 4.3 Managing Certificates and Certificate Validation for more complete information.

1.2.6 Connectivity Requirements

Communication between the TOE and the systems providing authentication and authorization services to the TOE must be protected from eavesdropping and modification.

The operational environment must assure that the TOE is the only interface (i.e., bridge) between the systems where the information flow policy has to be enforced by the TOE.

Any Network Time Protocol server the TOE uses to synchronize the realtime must be a reliable time source. Also, the realtime clock of the underlying operating system must provide reliable time stamps.

1.3 Obtaining and Verifying the TOE

The TOE guidance is also delivered as electronic downloads from IBM Fix Central. The TOE guidance is the IBM WebSphere DataPower documentation delivered as a standalone instance of IBM Knowledge Center.

In addition to the IBM Knowledge Center, the TOE guidance also includes the following documents. For these documents, the customer is required to contact IBM Support whose representative will provide instructions and the location to download from a secure FTP server.

- *DataPower Secure Deployment Guide (DataPower_Secure_Deployment_6020.pdf)*

1.3.1 Downloading from IBM Fix Central

To access IBM Fix Central, go to http://www.ibm.com/support/fixcentral. When you download from IBM Fix Central, you must the Download Director download method. Download Director provides the ability to view the status of the download in the poppup window and receive a notification if for any reason the download was not completely successful.

Note: IBM Fix Central shows fix sizes in Gigabytes (GB). Download Director uses bytes and 1024 bytes not 1000 bytes, in a Kilobytes (KB).

1. From the IBM Fix Central page, select the product and release.
   - In the **Product selector** field, enter **DataPower**.
   - From the results, select **6.0.2**.
   - Click **Continue**.
2. Identify the fixes to download.
   - In the **Text** field, enter `602`.
   - Click **Continue**.
3. Select the fixes to download.

- Select the firmware image for the product: XG45, XI52, or XB62.
- Select the IBM Knowledge Center archive.
- Click **Continue**.

4. If you did not login with your IBM Support credentials, login.
5. Choose the download method.
   - Select **Download using Download Director (requires Java enabled browser**.
   - Click **Continue**.
6. Agree to the terms and conditions
7. Click **Download now**.

Because the download method is Download Directory, the specified download directory on your system contains all images for the selected product and the IBM Knowledge Center archive for the DataPower documentation.

See section 1.3.6 for instructions to verify the integrity of the TOE firmware.

See section 1.3.2 for instructions to verify the integrity of the IBM Knowledge Center Documentation download.

1.3.2 Verifying the IBM Knowledge Center Download

The IBM Knowledge Center documentation download needs to be checked to verify that it is for the certified version of IBM WebSphere DataPower. The IBM Fix Central lists SHA-256 hash sum for it. If the hash sum of the download file matches that quoted on the IBM Fix Central, you can assume it is verified.

You can use any available sha256 utility to verify the checksum, for example, the sha226sum tool on Linux or Unix.

1.3.3 Installing the TOE Guidance (IBM Knowledge Center)

A downloadable IBM Knowledge Center contains the product documentation for IBM WebSphere DataPower SOA Appliances 6.0.2. You can download a customer-installable IBM Knowledge Center that contains the product documentation for IBM WebSphere DataPower SOA Appliances 6.0.2. The downloadable IBM Knowledge Center runs on a Windows operating system.

To install the TOE documentation, you can create the IBM Knowledge Center on you workstation or on a server where it can be accessed by others. The IBM Knowledge

Center image to install is on your system in the directory that you specified with the Download Directory download method.

1. Copy the IBM Knowledge Center archive to a location of your choice.
2. Extract the files in the archive to a location of your choice.
3. Read the `NOTICES.txt` file and the `README.TXT` file in the `\KnowledgeCenter` directory.

1.3.4 Starting and Stopping the IBM Knowledge Center Documentation

Start IBM Knowledge Center by running a batch file. View IBM Knowledge Center at the default URL. The first time that you start IBM Knowledge Center, you might have to wait a few minutes before the content is displayed.

1. Open a command prompt.
2. Change to the *`download_dir`*`\KnowledgeCenter\bin` directory.
3. Run the `startKC.bat` file.
4. View IBM Knowledge Center at this default URL: http://localhost:9090/kc/?lang=en

The command prompt window shows the status messages and the URL where IBM Knowledge Center is available.

Stop IBM Knowledge Center by running a batch file.

1. Open a command prompt.
2. Change to the *`download_dir`*`\KnowledgeCenter\bin` directory.
3. Run the `stopKC.bat` file.

The command prompt window shows the status messages.

1.3.5 Running IBM Knowledge Center on a Server

If you have a server with a static IP address, you can run IBM Knowledge Center from that server. Then, you can share the URL with other users in your organization.

This option is useful if your users are restricted to access only URLs within the organization intranet and are unable to access IBM Knowledge Center on the IBM website. After you start IBM Knowledge Center, you can provide users with the URL.

The URL has the following format, where *xxxx* is the port number:

```
http://fully-qualified-hostname-or-IP:xxxx/kc/?lang=en
```

See the following example URLs:

```
http://kcserver.example.com:9090/kc/?lang=en
http://192.0.2.0:9090/kc/?lang=en
```

1.3.6 Installing and Verifying the TOE Firmware

You must install the firmware image on the appliance. The firmware image to install is on your system in the directory that you specified with the Download Directory download method.

Place the downloaded firmware image on a server that offers http/s, scp or sftp access to the firmware image.

Log into the device as an admin with permissions in the default domain.

Enter flash mode

```
co;fl
```

Copy the firmware to the image: directory of the device.

```
Copy http://server_address:port/firmware image:///image.scrypt3
```

For more complete information about installing the firmware, Use the following installation tasks in the *Upgrade and rollback* container in IBM Knowledge Center.

- Transferring firmware images.
- Installing firmware images

You must verify that the device is in Common Criteria mode before booting the device. To see the current mode, execute the command `show system;` the product mode shows `cc`. If the appliance shows this mode, you must only reboot the appliance with the new firmware. Use a command of the following form:

```
boot image.scrypt3
```

If the appliance is not in Common Criteria mode, you must reinitialize the device. When you do this, all configuration data, and files placed on the TOE by an admin are lost; the device is returned to factory state. Because network configuration is lost, you must connect to the TOE using a physical serial connection. See Chapter 4 Setting up the initial configuration of the Secure Installation Guide for instructions.

If you want to preserve any configuration data, you must first copy the configuration files off of the device. As the device is not in Common Criteria mode and thus not in the evaluated configuration, you can use the WebGUI to copy the configuration files off of the device before reinitializing the device.

When you are ready, use the reinitialize command:

```
reinitialize firmware_image.scrypt3
```

When you can again log in to the TOE, the password for the admin account is admin. You will be prompted to change this. Then enter the `startup` command to begin the startup wizard. See the Secure Installation Guide for instructions.
To verify the integrity of the TOE firmware, following these instructions:

- No error should occur during installation of the evaluated firmware image. You should receive the login prompt.
- If an error has occurred, check the error message.
- If the firmware image has become corrupt in any way, the installation procedure will indicate that verification has not succeeded. The firmware is both signed and encrypted. In this case, you should again obtain the firmware from IBM FixCentral.
- If you have attempted to install the firmware on a non-7199 or non-7198 (9005) device, the error message will indicate you are using the wrong platform. Use a 7199 or 7198 (9005) device.
- If you upgrading the firmware from a version prior to 5.0.0.0, you must first upgrade the firmware installed on the device to 5.0.0.0 and then install the evaluated firmware version 6.0.2.0.

1.3.7 Confirming the Version of the TOE Firmware

To ensure that your appliance after initialization is in Common Criteria compatibility mode, ensure that the appliance is running a firmware version 6.0.2.0 at build 256732, the product mode is Common Criteria, and that the cryptographic hardware is disabled. You can validate these settings from the GUI or command line.

- If the product mode is not Common Criteria, you must reinitialize the appliance.
- If the cryptographic hardware is not disabled, you need to use the following commands and reboot the appliance.

```
# configure terminal
# crypto
# crypto-hw-disable all
# exit
# shutdown -reboot
```

Command line

- For the firmware version and build, you will see these value after login. You can also enter `show version` for this data.
- For the product mode, enter `show system`, the product mode shows `cc`. If the appliance does not show this mode, you must reinitialize the appliance.

- For the cryptographic hardware, enter show `crypto-hw-disable`. The values for `Current` and `Target` are `All`. If `Current` shows another value but `Pending Target` is `All`, reboot the appliance.
- For the cryptographic hardware, enter `show crypto-engine`. The values for `Disabled Hardware Feature` is `All`. If any other value, disable the cryptographic hardware with the commands above.

1.4 Configuration Requirements

The rest of this Guide describes configurations to be made to the TOE installation to comply with CC requirements. Below is a summary of the configuration information applies to the evaluated configuration:

- Audit must always be enabled
- For the establishment of a secure channel between the TOE and other IT entities, only TLS version 1.2 is allowed. SSL version 3.0 and TLS versions 1.0 and 1.1 must be disabled.
- The WebGUI for administrative management must be disabled
  Enter the command **co; no web-mgmt** to be sure it is disabled.
- SNMP must be disabled
  Enter the command **co; no snmp** to be sure it is disabled.
- The XML Management Interface must be disabled
  Enter the command **co; no xml-mgmt** to be sure it is disabled.
- The USB port cannot be used as the hardware is not active
- The use of CoProc is disallowed
  Enter the command **co; no xslcoproc** to be sure it is disabled.
- The use of IPMI is disallowed
- Enter the command **co; no ipmi-lan-channel mgmt0** to be sure it is disabled.

1.5 Guide to the Rest of the Document

Now that we've covered the necessary background, we will launch into a discussion of the TOE's "spheres of administration" and the types of administrators that the TOE supports. This section is called "Spheres of Administration and Administrators". The other sections are as follows:

- The section entitled "Initial Administrator Tasks" details the tasks the administrator must perform to configure the TOE correctly. This section also describes a number of optional tasks the administrator may perform.

- The section entitled “Conforming to the Evaluated Configuration” provides detailed information about how to conform to the configuration used for the evaluation.
- The section entitled “Application Domain Tasks” uses a common business scenario to explain how to set up services that can implement the policies desired.
- "Other Administrator Concepts and Tasks" provides important information on on-going administration of the TOE (that was not already covered in under Business Scenarios and Administrator Tasks). For example, this section discusses archiving of audit and configuration files.
- "Miscellaneous Information and Guidance" provides additional information and caveats that will help you maintain a secure configuration and do error resolution.
- Appendix on "Required Conditions for Network-Based Administration"
- Appendix on CLI commands included in the evaluated configuration

2 Spheres of Administration and Administrators

2.1 Application Domains and Application Domain Administrators

The fundamental unit of administration in the TOE is the "Application Domain" (AppDomain). At one level, an AppDomain is merely a portion of the TOE file system. It is a set of directories and files reserved for the creation and maintenance of firewalls or gateways that serve as proxies for a related set of backend services.

At another level, an AppDomain is an *environment.* When an AppDomain administrator logs into the TOE, they are immediately "placed" within their associated AppDomain. The AppDomain administrator can access all of the objects and files in their AppDomain. The AppDomain administrator also has access to the "temp" directory, to the "audit" directory, and to other TOE directories. To the AppDomain administrator, it appears that the entire TOE file system is at their disposal (modulo restrictions on audit log handling).

In actuality, the TOE is controlling the AppDomain administrator's view of the TOE file system, allowing access to "common" (system-wide) areas such as "audit", but disallowing access to other AppDomains. An AppDomain admin cannot even view a list of objects in other AppDomains.

Each AppDomain has its own configuration file. This stores the definitions of the firewalls and ancillary objects (such as log files) that comprise the AppDomain. This configuration file is "brought up" on startup of the TOE after the system configuration file is brought up. When a configuration file with a firewall definition is "brought up" by the TOE, the firewall itself is instantiated and ready to process data traffic.

It is through AppDomains that the TOE provides for separations of domains. AppDomains lets firewall proxies for separate backend services be administered separately and without interference.

An AppDomain administrator (AppDomain admin) has permission to create, modify, and delete objects and files within their AppDomain. However, in general, an AppDomain administrator cannot modify the objects in any other AppDomain.

2.2 The Application Domain File System

The following CC-relevant directories are "per domain", that is, each AppDomain has its own:

• config:	The TOE stores the domain configuration file here.
• temporary:	Available for use by the admin
• logtemp:	The TOE stores log files and rolled-over log files here.
• logstore:	Available for use by the admin.
• local:	Available for use by the admin

The following CC-relevant directories are system-wide, that is, there is a single directory for the entire TOE, and it is viewable (but not modifiable) by all administrators:

• audit:	TOE stores the audit file and the rolled-over audit file here.
• store:	This directory holds a number of built-in objects useful in firewall creation.
• image:	This directory holds the TOE boot image.

There are some additional directories that are viewable by an AppDomain administrator but which are not relevant to the TOE.

2.3 Privileged Administrators

Administrators that are created as "privileged" have access to all TOE commands and all administrator-visible resources. Only a privileged administrator (and not an AppDomain administrator) can set up and manage security-relevant network service settings such as the IP addresses of the TOE's physical interfaces. Only privileged administrators can set up AppDomains and AppDomain administrators, and only privileged administrators can control sensitive appliance operations such as shutdown.

While privileged administrators have full access to all the objects in all the application domains (and thus can create and modify services), the duties of the AppDomain administrators and the privileged administrators should be separate.

Note: The initial administrator of the TOE (the administrator named "admin") is a privileged administrator. If that administrator is you or you are otherwise a privileged administrator, read the rest of this document before creating either more privileged administrators or AppDomain administrators.

2.4 Summary of Application Domain Admins and Privileged Admins Duties

Here is a list of the main tasks of AppDomain administrators and privileged administrators:

AppDomain administrators:

1. Decide on the information flow policies needed for their backend applications.
2. Create and instantiate service objects that implement these policies.
3. Create ancillary objects such as log files that will allow for selective monitoring of message traffic and system events pertinent to the AppDomain.
4. Review their log files and the audit file
5. Archive the AppDomain's configuration file and log files.
6. Clean up temporary files in their AppDomain to free up disk space.

The CLI commands available to AppDomain admins are identified in the list of all TOE CLI commands given in Appendix B. The precise use of each command is explained in IBM Knowledge Center that accompanies this Guide.

Privileged administrators:

1. Configure network settings (for example, associate physical communications interfaces with IP addresses)
2. Create application domains
3. Create application domain administrators
4. Create other privileged administrators as needed to maintain continuity of administration
5. Perform other privileged appliance operations such as shutdown, clock and time zone setting, and other operations as needed
6. Review the system log and the audit file
7. Archive the audit log, the system log, the system startup configuration file and other desired files
8. Clean up space in system-wide temporary directories

All TOE CLI commands given in Appendix B are available to privileged administrators. The precise use of each command is explained in IBM Knowledge Center that accompanies this Guide.

3 Initial Administrator Tasks

This section will discuss the tasks the default privileged administrator must perform to create a configuration that conforms to the evaluated configuration. This section will also discuss actions the default privileged administrator can take as desired.

3.1 Required Tasks

Log into the TOE, if you haven't already done so:

```
login: admin
```

After you enter your account name, the TOE prompts for a password. It looks like this:

```
Password: <password>
```

Note that if you are "idle" for fifteen (15) minutes, you will have to login again.

After logging in, you must enter "global configuration mode" to perform the rest of the tasks discussed here. To do so, the administrator (admin) issues the command at the CLI prompt:

```
> configure terminal
```

3.1.1 Set the Time zone

Set the time zone as appropriate to your enterprise. The time zone for the local time affects the time that is displayed by the appliance. The appliance clock runs on Coordinated Universal Time.
Time zone settings are done with "subcommands" under "Timezone Configuration Mode". A "subcommand" is a CLI command that is only available in a particular configuration mode. To enter the configuration mode type:

```
> timezone
```

If you are in a standard zone, you need only use the "name" subcommand to set the time zone. The syntax is

> **name** *timezone-name*

When you use a standard time zone name — the list is provided in the CLI documentation for the time zone "name" subcommand and is provided below — then related time zone settings such as daylight savings time values are set for you. If you have a custom time zone, you must set the off-set from GMT and the daylight savings time values manually (using the "custom", "direction", "offset-*" and "daylight-*" subcommands of the Timezone Configuration Mode).

To set the time zone to Eastern Standard Time, use the following command:

Timezone > **name** EST5EDT

This is the time zone name for Eastern Standard Time.

Completes the time zone configuration by issuing the exit command

> **exit**

The **exit** command is the standard command to exit a command mode, thus completing the configuration created in that mode. If the admin made a mistake and didn't want to complete the configuration, instead enter the following command:

> **cancel**

Here are the names for the time zones:

HST10 Honolulu 10 hrs West of UTC, no DST
AKST9AKDT Alaska 9 hrs West, US DST rules
PST8PDT Pacific 8 hrs West, US DST rules
MST7MDT Mountain 7 hrs West, US DST rules
CST6CDT Central 6 hrs West, US DST rules
EST5EDT Eastern 5 hrs West, US DST rules
AST4ADT Atlantic 4 hrs West, Canada DST rules
UTC Universal Time UTC, no DST
GMT0BST GMT UTC, UK DST rules
CET-1CEST Central Europe 1 hr East, EU DST rules
EET-2EEST Eastern Europe 2 hrs East, EU DST rules
MKS-3MSD Moscow Time 3 hrs East, Russian DST rules

AST-3 Saudi Arabia 3 hrs East, no DST
KRT-5 Pakistan 5 hrs East, no DST
IST-5:30 India 5:30 hrs East, no DST
CST-8 China 8 hrs East, no DST
WST-8 Western Australia 8 hrs East, no DST
JST-9 Japan 9 hrs East, no DST
CST-9:30 Central Australia 9:30 hrs East, no DST
EST-10 Eastern Australia 10 hrs East, no DST

3.1.2 Set the Clock

Check the TOE's hardware clock and make any needed adjustments. First check the clock by issuing the following command:

> **show clock**

Now, if the time is incorrect, set the correct time by issuing the following command:

> **clock** { *yyyy-mm-dd* | *hh:mm:ss* }

Note that the TOE interprets the values for both date and time to be in the currently set time zone.

3.1.3 Set a Network Time Server

The TOE can use an NTP server to maintain time. Follow these steps to configure an NTP server.

> **ntp** *ntp_server_address*

The *ntp_server_address can* be in either IPv4 notation or IPv6 notation.

3.1.4 Set Cryptographic Hardware Mode

To comply with the Common Criteria guidelines, it is necessary to set the mode of operation for the cryptographic hardware. In the Global command mode, type the following sequence of commands:

```
> crypto
> crypto-hw-disable all
> exit
```

3.1.5 Block Connections to Services from Internal Addresses

The administrator must create an ACL to block internal addresses from connecting to the service (external) local-address. Here are the steps he must do:

> **acl** blockInternalAddresses

> **deny** *InternalNetworkSegment/SubnetMask* (for example 10.1.2.0/24)

> **allow** any

> **exit**

This ACL must then be applied to all Front Side Protocol Handlers and XML Firewalls created. This means this ACL must be created in each application domain as well as the default domain.

3.1.6 Configure the Audit Log

Administrators must take steps to configure the audit log so that no records are lost. This configuration depends upon the amount of traffic handled by the TOE and the frequency with which audit logs are retrieved.

The audit log is a text file maintained on the TOE's persistent storage. Only the router software can write to the log in any way; it is not directly modifiable by administrators. The audit log persists across reboots and reconfigurations. When the audit file **audit:///audit-log** reaches the size limit (the default is 1000 kilobytes) specified in the audit log settings, it is renamed to the backup name **audit:///audit-log.1** and a new file is opened as **audit:///audit-log.** This process is called log roll-over. The audit log will roll over as many times as specified in the audit log settings (the default is 3). When the audit log fills up after the first backup file has been created, the file audit://audit-log.1 is renamed to audit:///audit-log.2 and the current audit log is renamed to audit:///audit-log.1

and a new audit:///audit-log file begins again. This method continues until the rollover limit is achieved. When the limit of rollover files is reached, and the audit log must roll over again, the oldest rollover log file is overwritten. If no further disk space is available before the last rotation takes place, no further transactions are accepted until the administrator makes space.

On a machine accepting a high load of traffic, the administrator must set a combination of the size of the audit log (maximum is 500,000 kilobytes or 500 MB) and number of rotations (the maximum is 100) large enough to allow enough time to archive the oldest audit log file before it is overwritten or no further traffic is accepted.

To configure the audit log to use the maximum possible space and thus provide a guarantee that no logs are lost because traffic is no longer accepted, use the dir: command to determine the amount of free space on the TOE.

```
> co
> fl
> dir audit:
```

Divide the reported available space by 500 to arrive at the number of rotations needed.

Here is an example of the commands to configure the audit log.

```
> co
> audit-log-settings
> size 500000
> rotation 25
> exit
```

This set of commands makes the size of each log 500 megabytes and the number of rotations is set at 25, for a total of 1.25 gigabytes of audit log storage before a log is overwritten.

In addition, the TOE maintains an audit reserve, which is released when the disk is otherwise full. This reserve allows the audit log the space needed to capture logs from transactions already in process when no further transactions are accepted.

When the appliance forces the release, the log will contain a message that states that the disk space for audit events is low.

The audit reserve is 40 kilobytes by default. If more space is desired, the administrator can use the audit-reserve command to change it. Here is an example of that command increasing the reserve space to 100K.

```
> audit-reserve 100
```

Before restoring the appliance to service, a privileged administrator needs to free up disk space. When there is enough available disk space for normal operations, the administration can restart the appliance, which will resume the processing of traffic.

3.2 Optional Tasks

The administrator can perform these tasks as desired.

3.2.1 Set a Domain Name Server

Ahe TOE can use a Domain Name Server (DNS) to allow network addresses to be expressed using domain names rather than network addresses. Follow these steps to configure an NTP server.

```
> dns dns_server_address
```

The *dns_server_address can* be in either IPv4 notation or IPv6 notation.

3.2.2 Setting the Login Failure Threshold and Disablement Behavior

Once placed in Common Criteria mode during initialization, the TOE automatically enforces a login failure threshold, which is set to 3 failures. To reset the Login Failure Threshold, the admin issues:

```
> account max-login-failure allowableFailuresBeforeLockout
```

Note: A successful login by an administrator resets that administrator's login failure count to zero.

Example's admin issues the following command:

```
> account max-login-failure 4
```

The TOE's default behavior is to disable an account where the login threshold has been reached. Because this behavior poses a denial-of-service threat (where all accounts are disabled by a malicious person) the TOE includes the following command, which allows an administrator to set a time for disablement:

> **account lockout-duration** *minutesOfLockout*

For Example, an administrator who wants to enforce 8 hours of lockout, sets the duration with the following command:

> **account lockout-duration** 480

Please note that in the evaluated configuration, the lockout period for the admin account is 120 minutes by default before it is re-enabled by another administrator with the privileged administrator role.

If you wish to keep the "permanent disablement until reset" behavior, simply do not issue the lockout-duration subcommand. If you set a lockout-duration, and want to go back to "permanent disablement until reset" behavior issue the following command:

> **account lockout-duration** 0

In the evaluated configuration, the admin account cannot be permanently locked out. If the 'account lockout' is set to 0, the admin would still be able to login after 120 minutes.

3.2.3 Throttle Settings

The TOE automatically enforces throttling when the available memory of the TOE reaches a default threshold. The administrator can reset these thresholds and behaviors if desired.

The admin issues the following command:

> **throttle** *throttle-threshold-percent kill-threshold-percent minutes-till-timeout*

Example's admin issues the following command:

> **throttle** 10 3 15

The TOE enforces the throttle command:

- Refuses new client connections when only 10 percent of memory is available
- Shuts down the TOE if free memory goes down to 3 percent
- Shuts down the TOE if free memory has not exceeded 10% after 15 minutes

To disable throttling, the admin issues the following command:

> no throttle

3.2.4 Enabling Network-based Administration

See the caveats in **Decide on Enabling or Disabling Network-based Administration.**

Before enabling network-based administration, you must create an "ACL" (Access Control List) that restricts network-based administration the private LAN segment you have set up for this purpose.

To create an ACL for SSH, the Example administrator goes through a similar procedure:

> **acl** ssh

> **allow** *ip_addr_segment/subnet_mask* (for example 10.99.9.0/27)

> exit

Now, with client address controls for CLI network-based administration set up, the administrator can now enable the SSH servers on the TOE.

To enable SSH CLI administration, issue a command of the following form:
> ssh *ip_addr port_no*

For example:

> **ssh** 10.99.9.9   334

Note that the address set for SSH must be included in the network address segment allowed by the ACL.

Here, the SSH server is listening on the non-standard port number 334. There is a non-settable SSH "idle session" timer, after which the SSH connection is dropped.

Note that the TOE automatically associates the ACL named "ssh" with the SSH service.

The administrator of the default domain should capture the fingerprint of the DataPower ssh keys when first connecting to the device. Here is an example:

```
ssh-rsa 1024 73:be:93:97:42:b2:df:2d:50:e0:6e:7d:3a:0d:b0:ef
```

To ensure that any other administrators of any kind are connecting to the correct device, this fingerprint should be shared with all administrators. This fingerprint should then be compared to the fingerprint obtained from the device whenever an ssh connection is made.

3.2.5 Create an AppDomain

To create an AppDomain, the administrator must enter Application Domain Configuration Mode. The administrator issues a command of the following form:

> **domain** *domainName*

where domainName is the name of the application domain. domainName must be unique within the AppDomain namespace. The domain name can contain a maximum of 32 alphanumeric characters. Valid characters are:
“a” through “z” “_” (underscore)
“A” through “Z” “-” (dash)
“0” through “9”

For example:

> domain ConsumerInfoServices

This command creates the domain if it didn't exist, or accesses the existing named domain.

> **visible-domain** default

This command allows the domain to "see" directories in the default domain that otherwise would be inaccessible. The relevant directory is the store: directory.

Administrators do not generally need to access this directory explicitly, however the TOE needs to access certain files in store: for message processing.

As usual, to save these changes the admin enters:

> exit

The TOE now creates the directory structure needed for the AppDomain.

There is an additional step required so that the domain becomes part of the TOE's stable configuration.

> **write mem**

This step causes the domain configuration to be become part of the TOE's "startup configuration" and hence be stable across reboots.

If the admin wants to delete an AppDomain at some point, the admin must issue the following commands:

> **no domain** *domainName*

> **write mem**

Repeat these steps to create any desired additional application domains.

3.2.6 Create the AppDomain Administrators

To create an administrator, you enter User Configuration Mode as follows:

> username *account-name*

For example:

> username Peter

Next, to set the password:

> password *password_string_for_Peter*

To create Peter as an AppDomain admin for the ConsumerInformationServices domain, the admin does two steps:

> **access-level** privileged

> **domain** ConsumerInformationServices

Note: All admins in the TOE have the access-level designation of "privileged". It is the domain subcommand that restricts the admin account to the specified domain or domains. Without the domain subcommand, the admin is created as privileged.

To complete account creation:

> exit

Unlike domains and other objects in the TOE, user accounts are persistent as soon as they are created. There is no need to issue the "write mem" command to cause an account to become part of the startup configuration.

Notes on passwords:
The TOE does not show a password example because the password string is not echoed.

Note that in order to promote non-obviousness and to increase the password search space, the TOE mandates the following password policy:

- minimum of 14 characters in length,
- contains one lower character and one uppercase character,
- contains one number,
- contains one special character,
- must be different from the most recent N pass password, where N is a configurable value that shall be greater or equal to 3.

Also the , TOE mandatory password policy is to allow only printable characters and to allow a maximum of 128 characters.
Additionally, the TOE optionally enforces the following:

- the password cannot contain the user name (pwd-username subcommand in rbm mode). In CC mode, the default value it is enforced.

- the users must change periodically their passwords (pwd-aging and pwd-max-age subcommands in rbm mode). In CC mode, password change is enforced with a frequency of 90 days.

Passwords must be kept confidential and should not relate to the administrator's personal information in any obvious way.

***

Do not "share" a single administrator account among more than one person. To maintain accountability each administrator must have their own account.

3.2.7 Saving Error Information

If the TOE crashes, post-mortem diagnostic information can be recovered and sent to IBM Support to help diagnose the cause of the failure. In the unlikely event of such a crash, a privileged administer should log on and from the default domain issue the following command:

> **save error-report**

This command creates an error report on the TOE named temporary:///error-report.txt in the default domain. This file can then be copied off-host, examined by an administrator, and provided to IBM Support. (See the discussion under "Backup and Restore of Configuration Files and Other Files" for instructions on copying the file.)

The report includes the audit logs, the system logs from each domain, and the saved configuration information for the default domain.

The administrator might also wish to save configuration information for each domain.

The way to do this is:

> **switch domain** *domain*

If you wish to have a record of the running configuration, you must first save the configuration:

>**write mem**

Now, copy the configuration file for the domain off-box. Its name is config:<domain>.cfg. For example, for the SupplierLogistics domain, the config file is config:SupplierLogistics.cfg .

3.2.8 Saving the In-Memory Configuration

When you make changes to the TOE, these are stored in working memory. If you want the changes to persist across reboots, you must save the changes with the **write memory** command. To do so you issue the following command:

> **write mem**

The TOE prompts if you wish to overwrite the existing autoconfig.cfg file. Type 'y' or 'yes'.

Overwrite previously saved configuration [y/n]? yes

Note that the running configuration of the "default domain" — which is the privileged system domain — will be saved in ***config:///autoconfig.cfg***. There is no way to change what file the running configuration gets saved to.

The first time you save memory, you will want to have ***config:///autoconfig.cfg*** be the system startup file. Later on, after other changes to the system, you might want to save the running configuration but have a prior configuration designated as the startup file. We discuss how to do this under "Designating the System Startup File".

Note that "write mem" only saves the configuration of the domain that it is issued in. Each domain has its own startup file. This means that AppDomain administrators will also issue "write mem" as needed to save the working configuration of their domain (for example, firewall definitions).

Note: If you have not already done so, issue "write mem" to save the changes you have made.

4 Conforming to the Evaluated Configuration

Administrators must be sure to follow the guidelines given here in order to create configurations conformant with the evaluated configurations of the product.

4.1 Administrative Interfaces

In the evaluated configuration, only the CLI interface accessed through the serial console or the SSH terminal service is allowed. All other administrative interfaces (i.e., WebGUI, SNMP, and XML Management) must not be enabled.

4.1.1 Verifying the TOE is running in Common Criteria (CC) mode

Common Criteria (CC) mode puts the DataPower TOE appliance in a mode that enforces a set of policies defined by the CC certification. To verify that the TOE is running in the CC mode, issue the following the CLI command: `show services` which will displays 'product mode: cc'.

4.2 Services

Administrators can use an XML Firewall (on XI52 and XG45 only), a Multi-Protocol Gateway or a Web Service Proxy to build services. Use of any other service, such as an XSL Proxy, is outside the boundary of the evaluated configuration.

4.3 Managing Certificates and Certificate Validation

Administrators must take care to observe the following guidelines:

- The administrator must take care not to use certificates containing an MD5 signature nor certificates derived from a Certificate Authority (CA) certificate containing an MD5 signature.
- All certificates used must employ an exponent of 65537 (rather than 3).

- Administrators must ensure that the minimum key size for the RSA certificates is 1024 bits, recommended would be key sizes longer than 1024 bits. Note that this applies to keys contained in the certificate.
- When generating RSA key pairs externally and importing them into the TOE, administrators must ensure that the private keys have CRT (Chinese Remainder Theorem) format (the default in OpenSSL). This means the private RSA keys must have the parameters required for CRT.
- When importing certificates into the TOE, administrators must ensure that the certificates do not include regular expressions (like *.com) or other bogus expressions. In addition admins should ensure that the certificate is signed with a key of a key size that is recommended to be even longer than the size of the key contained in the certificate.
- The administrator must take care to maintain the certificates stored on the device, updating or deleting those that have expired.

This refers to the certificates employed for SSL/TLS connections, the certificates employed in a Sign or Encrypt action, and the certificates employed in a Validation Credential.

When using a Validation Credential (such as used to verify signing certificates), the certificate validation mode must be PKIX and the chain of certificates must be present in the list of certificates included in the Validation Credential.
Any validation of certificates must use a Certificate Revocation List to ensure certificates used by the TOE have not been revoked. Here is an example of the commands to create a CRL:

```
> crl HTTP-Fetch http
crl> fetch-url http://crl.verisign.com/ATTClass1Individual.crl
crl> exit
> write mem
```

See the Knowledge Center for more information about these commands.

4.4 Transport Protocols

The evaluated configuration is restricted to the following Front Side Protocol Handlers:

HTTP
HTTPS

SFTP Poller
SFTP Server
FTP Poller
FTP Server

Use of any other Front Side Protocol Handlers is outside the evaluated configuration. The use of GSKit as a key repository is outside the boundaries of the evaluated configuration.

4.4.1 TLS

In the evaluated configuration, only version 1.2 of TLS is allowed. TLS is used for the establishment of a secure session for the HTTPS and FTP over TLS protocols.

The evaluated configuration allows the following:
Ciphersuites:
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256

Cryptographic key generation algorithms and key sizes:
- AES keys with 128 and 256 bits
- Triple-DES keys with168 bits
- HMAC keys with 160 and 256 bits

Encryption algorithms:
- RSAES-PKCS1-v1_5 with key sizes 1024, 2048, and 4096 bits
- AES (CBC mode) with key sizes 128 and 256 bits
- Triple-DES (CBC mode) with key size 168 bits

Message Authentication Code algorithms:
- MAC-SHA-1 and HMAC-SHA-1 (160 bits)
- MAC-SHA-256 and HMAC-SHA-256 (256 bits)

Digital signature verification and generation algorithms and key sizes
- RSASSA-PKCS1-v1_5 with SHA-1 and SHA-256: 1024, 2048, and 4096 bits

Please note that the evaluated configuration allows any combination of the above ciphers but disallows any ciphers that are not listed above.

4.4.2 SSH

In the evaluated configuration, only version 2.0 of SSH is allowed. SSH is used for the establishment of a secure session for SFTP server and poller handlers, SCP commands invoked from the command line interface and remote sessions started by administrators.

The evaluated configuration allows the following:
Cryptographic key generation algorithms and key sizes:

- AES keys with 128, 192, and 256 bits
- Triple-DES keys witth 168 bits
- HMAC keys with 160 bits

Encryption algorithms:

- AES (CBC and CTR modes) with key sizes 128, 192, and 256 bits
- Triple-DES (CBC mode) with key size 168 bits

Message Authentication Code algorithms:

- HMAC-SHA-1 (160 bits)

Digital signature verification and generation algorithms and key sizes

- DSA with SHA-1 (L=1024 bits, N=160 bits)
- RSASSA-PKCS1-v1_5 with SHA-1 and SHA-256: 1024, 2048, and 4096 bits

Please note that the evaluated configuration allows any combination of the above ciphers but disallows any cipher that are not listed above.

Administrators must ensure that the minimum key size for public keys used for server authentication is 1024 bits, recommended are key sizes longer than 1024 bits. If public keys are used for client authentication the same holds true.

4.4.3 Supported Protocols for Information Flow Control

All traffic through the TOE is subject to information flow policies. The TOE allows in the evaluated configuration the definition of rules that filter traffic based on the following information object group or a combination of them:

- IP source address
- IP destination address
- TCP port
- Request URL
- Message content
- XML Schema Definition
- XML signature

- Identity assert obtained from XML standards-based messages or transport layer information
- Protocol header (HTTP, HTTPS, FTP, FTP over SSL, SFTP) attributes
- Message size

Information flow control policy can be enforced based on the following object groups:

- services: Multi-Protocol Gatway, Web Services Proxy and XML Firewall.
- Front side protocol handlers: HTTP, HTTPS, FTP, SFTP.
- Access Control Lists: allows or denies IP addresses.
- Authentication, Authorization, and Audit policies.
- Application Security and Processing Policies: Matching Rules, Web Request and Web Respond Profiles, Processing Rules, URL Rewrite Policies.
- Cryptography: SSL Proxy Profiles, Crypto Profiles, Identification Credentials, Certificate and Key Aliases, digital certificates and private keys.

For application protocols supported by the TOE (HTTP, HTTPS, FTP, FTP over SSL, SFTP), the TOE denies any access or service requests that do not conform to its associated protocol specification (e.g., RFC).

The TOE can reject XML and SOAP messages based on schema validation. In addition, the TOE can perform XML signature generation and validation, XML encryption, and XML canonicalization based on [W3CXMSIG], [W3CXMLENC], [W3CXMC14N], and [W3CXMLEXCC14N].

For more information, please refer to the user guides available at the IBM WebSphere Knowledge Center.

4.5 AAA

Administrators can use any of the following methods to extract an identity from a request within an AAA Policy. Use of any other methods not specified here is disallowed in the evaluated configuration.

- HTTP Basic Authentication
- Subject DN from SSL client connections
- LTPA token
- WS-Security tokens: Username, Binary Security Token (X.509), or SAML (versions 1.0, 1.1 and 2.0)

The use of these methods requires the correct configuration of a AAA Policy. Here is an example of the commands to use the HTTP Basic Authentication method:

```
>co
>aaapolicy example-cc
aaapolicy>extract-identity
aaapolicy/extract-identity>method 1 http-basic-auth
aaapolicy/extract-identity>exit
```

Here are the AAA Policy commands to use the Subject DN from SSL client connections:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>method client-ssl
aaapolicy/extract-identity>exit
```

Here are the AAA Policy commands to use the Username/Password from a WS-Security token:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>method wssec-username
aaapolicy/extract-identity>exit
```

The AAA Policy may then be configured to use the desired authentication methods appropriate for the extracted identity. These are LDAP or an AAA Info file.

Here is an example of the AAA Policy commands to use an AAA Info file for authentication:

```
aaapolicy>authenticate
aaapolicy/authenticate>method xmlfile
aaapolicy/authenticate>xmlfile-url local:///aaainfo.xml
aaapolicy/authenticate>exit
```

Here is an example of the AAA Policy commands to use an LDAP server for authentication:

```
aaapolicy>authenticate
aaapolicy/authenticate>method ldap
aaapolicy/authenticate>remote-host 10.9.1.1
aaapolicy/authenticate>remote-port 389
aaapolicy/authenticate>ldap-bind-dn admin
aaapolicy/authenticate>ldap-bind-password password
aaapolicy/authenticate>exit
```

Here are the AAA Policy commands to use an LTPA token:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>method ltpa
aaapolicy/extract-identity>exit
```

The Authentication method used must then also be LTPA token.

```
aaapolicy>authenticate
aaapolicy/authenticate>method ltpa
aaapolicy/authenticate>ltpa-key cert:///ltpakey.key
aaapolicy/authenticate>ltpa-password huggies
aaapolicy/authenticate>exit
```

Here are the commands to use an X.509 public key certificate contained in a WS-Security Binary Security Token:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>method wssec-binary-token
aaapolicy/extract-identity>exit
```

The Authentication method can then be validation of the certificate. Here are the commands for this authentication method:

```
aaapolicy>authenticate
aaapolicy/authenticate>method binarytokenx509
aaapolicy/authenticate>x509-bin-token-valcred myvalcred
aaapolicy/authenticate>exit
```

Here are the AAA Policy commands to use the subject name of a SAML Authentication assertion to extract an identity:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>saml-authen-name
aaapolicy/extract-identity>exit
```

Here are the AAA Policy commands to use the subject name of a SAML Attribute assertion to extract an identity:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>saml-attr-name
aaapolicy/extract-identity>exit
```

Authentication using a Subject from a SAML assertion may use any desired method. Authentication using SAML may also be performed by validating the signature of a signed SAML assertion. Here are the commands:

```
aaapolicy>authenticate
aaapolicy/authenticate>method saml-signature
aaapolicy/authenticate>exit
```

Here are the AAA Policy commands to use a SAML Artifact as an identity:

```
aaapolicy>extract-identity
aaapolicy/extract-identity>saml-artifact
aaapolicy/extract-identity>exit
```

When using a SAML Artifact, authentication is performed using a remote artifact responder. Here are the commands:

```
aaapolicy>authenticate
aaapolicy/authenticate>method saml-artifact
aaapolicy/authenticate>saml-artifact-responder http://10.10.1.1:300/responder
aaapolicy/authenticate>exit
```

4.6 Use of XML stylesheets

Administrators can use custom stylesheets to transform, route or filter messages. A custom stylesheet can contain any standard XSLT 1.0 expression, but usage of the DataPower extension functions is not allowed in the evaluated configuration.

For performing transformations that include cryptographic functions like sign, verification, encryption and decryption of XML messages, DataPower provides a set of stylesheets shown in Appendix C: Cryptographic Stylesheets. In the evaluated configuration, these stylesheets cannot be modified and custom stylesheets cannot be used for this purpose (cryptographic functionality is implemented through extension functions).

The following sections show examples of how to use these stylesheets.

4.6.1 Sign

In order to create XML signatures with sufficient strength, the Sign action must use one of the following settings:

- HMAC-SHA-1 (160 bits, minimum 80 bits)
- DSA with SHA-1 with keys sizes L=1024 bits, N=160 bits
- RSASSA-PKCS1-v1_5 with SHA-1, SHA-256, SHA-384 and SHA512 with key sizes 1024, 2048, and 4096 bits

Here is an example of the commands that are needed to create a Sign action using WS-Security formatted XML and approved RSA keys and hash algorithms:

```
action "Policy_rule_0_sign_0"
  reset
  type xform
  input "INPUT"
  transform "store:///sign-wssec.xsl"
  output "PIPE"
  parameter "{http://www.datapower.com/param/config}hashalg" "sha384"
  parameter "{http://www.datapower.com/param/config}keypair-cert" "example-cert"
  parameter "{http://www.datapower.com/param/config}keypair-key" "example-key"
  parameter "{http://www.datapower.com/param/config}sigalg" "rsa"
  output-type default
exit
```

If you want to control the canonicalization algorithm as well, you can add the “c14nalg” parameter to the commands. The following example also demonstrates the use of different approved hash algorithms and signature algorithms:

```
action "example-cc_rule_1_sign_1"
  reset
  type xform
  input "INPUT"
  transform "store:///sign-wssec.xsl"
  output "OUTPUT"
  parameter "{http://www.datapower.com/param/config}c14nalg" "exc-c14n"
  parameter "{http://www.datapower.com/param/config}hashalg" "sha256"
  parameter "{http://www.datapower.com/param/config}keypair-cert" "example-cc"
  parameter "{http://www.datapower.com/param/config}keypair-key" "example-cc"
  parameter "{http://www.datapower.com/param/config}sigalg" "rsa-sha256"
  output-type default
exit
```

Use the store:///sign-soapsec.xsl stylesheet to sign XML files using the SOAP signature standard rather than the WS-Security standard. The commands are otherwise the same, as shown in this example.

```
action "example-cc_rule_1_sign_1"
  reset
  type xform
  input "INPUT"
  transform "store:///sign-soapsec.xsl"
  output "OUTPUT"
  parameter "{http://www.datapower.com/param/config}c14nalg" "exc-c14n"
  parameter "{http://www.datapower.com/param/config}hashalg" "sha256"
  parameter "{http://www.datapower.com/param/config}keypair-cert" "example-cc"
  parameter "{http://www.datapower.com/param/config}keypair-key" "example-cc"
  parameter "{http://www.datapower.com/param/config}sigalg" "rsa-sha256"
  output-type default
exit
```

Use the store:///sign-enveloped.xsl stylesheet to create enveloped signatures in any XML-formatted payload (not necessarily SOAP-formatted).

```
action "example-cc_rule_1_sign_1"
  reset
  type xform
  input "INPUT"
  transform "store:///sign-enveloped.xsl"
  output "OUTPUT"
  parameter
"{http://www.datapower.com/param/config}XPath" "/*[local-name()='statement']/*[local-
name()='charges']"
  parameter "{http://www.datapower.com/param/config}c14nalg" "exc-c14n"
  parameter "{http://www.datapower.com/param/config}hashalg" "sha256"
  parameter "{http://www.datapower.com/param/config}keypair-cert" "example-cc"
  parameter "{http://www.datapower.com/param/config}keypair-key" "example-cc"
  parameter "{http://www.datapower.com/param/config}sigalg" "rsa-sha256"
  output-type default
exit
```

Note that the Xpath parameter identifies a node within the document to be signed. The signature node then appears as a first child of that node. If you do not explicitly identify a node to sign that the entire message will be signed, and the signature node is placed as a first child of the root node of the document.

Use the store:///sign-enveloping.xsl stylesheet to create a signature that envelops the entire message within an Object node of the signature block.

```
action "example-cc_rule_1_sign_1"
  reset
  type xform
  input "INPUT"
  transform "store:///sign-enveloping.xsl"
  output "OUTPUT"
  parameter "{http://www.datapower.com/param/config}c14nalg" "exc-c14n"
  parameter "{http://www.datapower.com/param/config}hashalg" "sha256"
  parameter "{http://www.datapower.com/param/config}keypair-cert" "example-cc"
  parameter "{http://www.datapower.com/param/config}keypair-key" "example-cc"
  parameter "{http://www.datapower.com/param/config}sigalg" "rsa-sha256"
  output-type default
exit
```

4.6.2 Verify

The Verify action must use one of the following settings:

- HMAC-SHA-1 (160 bits, minimum 80 bits)
- DSA with SHA-1 with keys sizes L=1024 bits, N=160 bits
- RSASSA-PKCS1-v1_5 with SHA-1, SHA-256, SHA-384 and SHA512 with key sizes 1024, 2048, and 4096

A Verify action verifies signatures found in messages. Here are the commands to create a basic default Verify action:

```
action "example-cc_rule_1_verify_0"
  reset
  type filter
  input "dpvar_2"
  transform "store:///verify.xsl"
exit
```

You can control the signature type to verify only RSA/DSA signatures (the default), only HMAC signatures or any signature by adding the signature-method-type parameter:

```
action "example-cc_rule_1_verify_0"
  reset
  type filter
  input "dpvar_2"
  transform "store:///verify.xsl"
  parameter "{http://www.datapower.com/param/config}signature-method-type" "verify-all"
exit
```

You can restrict the signature algorithm used to RSA SHA1 for all RSA signatures by adding the restrict-algorithm parameter:

```
action "example-cc_rule_1_verify_0"
  reset
  type filter
  input "dpvar_2"
  transform "store:///verify.xsl"
  parameter "{http://www.datapower.com/param/config}restrict-algorithm" "on"
  parameter "{http://www.datapower.com/param/config}signature-method-type" "verify-all"
exit
```

In order to prevent passing signatures that use weak encryption or hashing functions (see below for a list of weak encryption and hashing functions), administrators must insert a Filter action into the MultiStep Processing Policy either before or after the Sign action. This Filter action must use a copy of the stylesheet store:///reject-weak-signatures.xsl. Use the following CLI commands to create a copy of the required style sheet:

```
co
flash
copy store:///reject-weak-signatures.xsl local:///reject-weak-signatures.xsl
exit
```

The Filter action must then use the copy placed in the local:/// directory. Here is an example of the commands that create a filter action:

```
action "Policy_rule_0_filter_0"
  reset
  type filter
  input "INPUT"
  transform "local:///reject-weak-signatures.xsl"
exit
```

Note that the Verify action does provide an option to use an Xpath expression to identify the node(s) signed, as one measure to combat XML Signature Wrapping attacks (XSW). Some forms of attack may still be possible.

4.6.2.1 Weak Encryption and Hashing Functions

Use of weak encryption and hashing functions for signatures are disallowed in the evaluated configuration. The following must not be used in the evaluated configuration function:

- URI-DIGEST-RIPEMD160
- URI-DIGEST-MD5
- URI-SIGN-RSA-MD5
- URI-SIGN-RSA-RIPEMD160
- URI-SIGN-RSA-RIPEMD160-2010
- URI-SIGN-HMAC-MD5
- URI-SIGN-HMAC-RIPEMD160

4.6.3 Encrypt

Administrators must use the following algorithms for encryption in the evaluated configuration. Algorithms not listed here are disallowed in the evaluated configuration.

Symmetric encryption:

- Triple-DES with key size 168 bits
- AES with key sizes 128, 192, and 256 bits

Asymmetric key transport:

- RSAES-PKCS1-v1_5 and RSAES-OEAP with key sizes 1024, 2048, and 4096 bits

DataPower implementations of TLS and XML encryption with the TDES algorithm do not change the TDES key bundle after 2^32 64-bit data blocks (32 GB). If customers want to encrypt documents longer than 32 GB or send more than 32 GB of TLS traffic, they should use one of the AES ciphers rather then TDES. Please note that using AES-CBC does not provide integrity protection. Use a signature for that assurance.

Here is an example of the commands that configure an Encrypt action to encrypt a standard XML-formatted file using AES128-CBC encryption:

```
action "Policy_rule_0_encrypt_0"
  reset
  type xform
  input "PIPE"
  transform "store:///encrypt.xsl"
  output "PIPE"
 parameter "{http://www.datapower.com/param/config}algorithm" "http://www.w3.org/2001/04/xml
enc#aes128-cbc"
  parameter "{http://www.datapower.com/param/config}recipient" "example-cert"
output-type default
exit
```

Note that these commands must be typed on the same line. Here is an example of the commands that configure an Encrypt action to encrypt a standard SOAP-formatted file ac cording to the WS-Security standard using AES128-CBC encryption and RSA keys for transport of the encryption key:

```
action "Policy_rule_0_encrypt_0"
  reset
  type xform
  input "PIPE"
  transform "store:///encrypt-wssec.xsl"
  output "PIPE"
  parameter "{http://www.datapower.com/param/config}algorithm"
"http://www.w3.org/2001/04/xmlenc#aes192-cbc"
  parameter "{http://www.datapower.com/param/config}encryption-key-type" "asymmetric"
 parameter "{http://www.datapower.com/param/config}key-transport-algorithm" "http://www.w3.o
rg/2001/04/xmlenc#rsa-1_5"
  parameter "{http://www.datapower.com/param/config}recipient" "example"
  output-type default
exit
```

Here is an example of the commands that configure an Encrypt action to encrypt a standard SOAP-formatted file according to the XML Security standard using AES128-CBC encryption and RSA keys for transport of the encryption key:

```
action "Policy_rule_0_encrypt_0"
  reset
  type xform
  input "PIPE"
  transform "store:///encrypt-soap.xsl"
  output "PIPE"
  parameter "{http://www.datapower.com/param/config}algorithm"
"http://www.w3.org/2001/04/xmlenc#aes192-cbc"
  parameter "{http://www.datapower.com/param/config}encryption-key-type" "asymmetric"
 parameter "{http://www.datapower.com/param/config}key-transport-algorithm" "http://www.w3.o
rg/2001/04/xmlenc#rsa-1_5"
  parameter "{http://www.datapower.com/param/config}recipient" "example"
  output-type default
exit
```

To encrypt only selected fields of the message, you must use a Document Crypto Map, which identifies the nodes to encrypt. Here is an example configuration encrypting only a selected node of an XML-formatted message.

```
document-crypto-map "example-cc"
  select "/*[local-name()='statement']/*[local-name()='charges']"
exit

action "Policy_rule_0_encrypt_0"
  reset
  type xform
  input "PIPE"
  transform "store:///encrypt.xsl"
  dynamic-stylesheet example-cc
  output "PIPE"
 parameter "{http://www.datapower.com/param/config}algorithm" "http://www.w3.org/2001/04/xml
enc#aes128-cbc"
  parameter "{http://www.datapower.com/param/config}recipient" "example-cert"
output-type default
exit
```

In all cases, substitute the following to use RSA-OEAP instead for the asymmetric transport key:

```
parameter "{http://www.datapower.com/param/config}key-transport-algorithm" "http://www.w3.or
g/2001/04/xmlenc#rsa-oaep-mgf1p"
```

4.6.4 Database Connectivity

Use of the SQL action to establish connections to remote database servers is outside the boundaries of the evaluated configuration.

4.7 SSL/TLS

In the evaluated configuration, administrators can only use the allowable ciphers and hash functions stated in section 4.3.1 when creating the Crypto Profile used by SSL Proxy Profiles for either client or server use.

Any Crypto Profile must disable all SSL and TLS versions except TLS v1.2. For that purpose, the options parameter must be as follows:

```
OpenSSL-default+Disable-SSLv2+Disable-SSLv3+Disable-TLSv1+Disable-
TLSv1d1
```

In addition, the cipher parameter must specify the symmetric encryption and hashing algorithms allowed; the following table shows the keyword used for each of the allowed cipher suites:

Cipher Suite	ciphers parameter
`TLS_RSA_WITH_3DES_EDE_CBC_SHA`	`DES-CBC3-SHA`
`TLS_RSA_WITH_AES_128_CBC_SHA`	`AES128-SHA`
`TLS_RSA_WITH_AES_256_CBC_SHA`	`AES256-SHA`
`TLS_RSA_WITH_AES_128_CBC_SHA256`	
`TLS_RSA_WITH_AES_256_CBC_SHA256`	`AES256-SHA256`

For allowing more than one cipher suites, use the plus sign to concatenate the possible ciphers. The keyword @STRENGTH can be used at any point to sort the current cipher list in order of encryption algorithm key length. Here is an example Crypto Profile configuration command that meets all the necessary criteria:

```
crypto
  certificate "example-cert" "cert:///dpower-sscert.pem"
exit

crypto
  key "example-key" "cert:///dpower-privkey.pem"
exit

crypto
  idcred "example-cred" "example-key" "example-cert"
exit

crypto
  profile "example-prof" "example-cred" option-string OpenSSL-default+Disable-SSLv2+Disable-
SSLv3+Disable-TLSv1+Disable-TLSv1d1 ssl example ciphers "AES256-SHA256:AES128-
SHA256:AES256-SHA:AES128-SHA:DES-CBC3-SHA@STRENGTH"
exit
```

TLS Compression

Please not that in the evaluated configuration, the use of TLS Compression is disallowed.

TLS Session Caching

If TLS session caching is desired, it can be configured by using the settings client-cache "on" client-sess-timeout "300" client-cahe-size "100" (the router defaults).
Example:

```
sslproxy "example-cc" "forward" "example-cc" client-cache "on" client-sess-timeout "300" client-cache-size "100"
```

If TLS session caching is not desired, set client-cache "off" with no further arguments.

4.8 Using Public Key Authentication for SFTP Poller

Administrators can use an SFTP Poller Front Side Protocol Handler to retrieve files from remote SFTP servers.

To use such a Front Side Protocol Handler, it is necessary to create an SSH Client Profile. Here is an example creating such an object:

```
> sshclientprofile "example-cc"
sshclientprofile>  user-name "hondo"
sshclientprofile>  user-auth "publickey+password"
sshclientprofile>  user-private-key example-cc-ssh
sshclientprofile>  user-password "password"
sshclientprofile>  strict-host-key-checking
sshclientprofile>  client-known-host 10.97.111.108 ssh-rsa AAAB3NzaC1yc2EAAAABIwAAAIEA1J/99rRvdZm
VvkaKvcG2a+PeCm25p8OJl87SA6mtFxudA2ME6n3lcXEakpQ8KFTpPbBXt+yDKNFR9gNHIfRlUDho1HAN/
a0gEsvrnDY5wKrTcRHrqDc/x0buPzbsEmXi0lud5Pl7+BXQVpPbyVujoHINCrx0k/z7Qpkozb4qZd8==
sshclientprofile> exit
```

To use only password authentication, omit "publickey+" from the user-auth command. To use only public key authentication, omit "password" from the user-auth command. It is not then necessary to use the user-password command.

The private SSH key used to identify the client is a Crypto Key object, which in turn allows the administrator to provide the necessary key material.

```
crypto
  key "example-cc-ssh" "cert:///dpower-privkey.pem"
exit
```

If strict host key checking is enabled, as shown in the example, then at least one client-known-host entry must exist, or the client will fail to establish any connection. If strict host key checking is not enabled (by not issuing the strict-host-key-checking command), then the client will automatically add the keys to each host to which it connects to the known-host table.

To view the entries in the known hosts table for the SSH Clients in the current domain, use the following command:

```
> show client-known-hosts

Host            Type          Client Name
10.97.111.108   ssh-rsa       sshClient1
```

To remove an entry from the client-known-hosts table, use a command similar to the following:

```
no client-known-host 10.97.111.108 sshClient1
```

Here is an example set of commands to create an SFTP Poller Front Side Handler

```
> source-sftp-poller "example-cc"
source-sftp-poller>  target-dir "sftp://sftpsvr.host.com:22/Files/"
source-sftp-poller>  match-pattern (.*).xml$
source-sftp-poller>  result-name-pattern $1.result
source-sftp-poller>  ssh-client-connection example-cc
source-sftp-poller>  exit
```

Note: The key generation of the default key is not part of the TOE. SSH key generation is not part of the TOE either. The administrator is advised to import keys that are externally generated or use the ones generated, however, the generation of these keys are outside the scope of this evaluation.

4.9 Using Public Key Authentication for SFTP Server

Administrators can use an SFTP Server Front Side Protocol Handler to accept incoming messages from remote SFTP clients.

The SFTP Server Front Side Protocol Handler includes a Host Private Keys property that identifies the SSH private key to use during session initiation. The property refers to a

Crypto Key object, that in turn identifies the key material itself. If this property is not used, the SFTP Server Front Side Handler uses the default SSH keys shipped with the device. These keys cannot be changed.
Administrators can use Public Key authentication to authenticate requests. An AAA Policy is used for this purpose.

The AAA Policy must use Processing Metadata as the method used to extract the identity of the remote client. The administrator must select ssh-password-metadata from the list of available metadata.

The AAA Policy must use a custom style sheet to perform the Authenticate step of the AAA Policy. The custom style sheet is provided in the default store:// directory of the product. It is store:///ssh-client-auth.xsl. This stylesheet is written to find and use either RSA or DSA keys. The TOE is shipped with default RSA stylesheet and DSA stylesheet.

This file opens and reads another file, store:///ssh-client-info.xml. Administrators must edit this file to place the public keys of requesting clients in the file. The public key presented by a remote SFTP client is compared to the keys found in this file. If there is a match, the client is authenticated; if not, the client is not authenticated.

Here is the section of the file administrators must modify:

```
  <sshclients>
    <client>
      <username>USERNAME</username>
      <pubkey>PUBLIC KEY BLOB HERE</pubkey>
    </client>
  </sshclients>
```

Administrators must create one <client> node for each client to authenticate. Within the client node, administrators replace USERNAME with the actual name of a remote client. Administrators replace PUBLIC KEY BLOB HERE with the actual public key presented by the given user. Both RSA and DSA keys work here.

Here is an example of a modified entry:

```
<sshclients>
     <client>
          <username>user_1</username>
          <pubkey>AAAAB3NzaC1yc2EAAAABJQAAAIBvG4cFb5GLoDf3Al
          W68ngfP0PpoXQEaDJtEuXXbXEBdr2uGjZ3FjCPaKlYOXaqCQKh
          q8Nn30ex0C1PtHtfwfw9p6yf/+xi5KYF7Fnm/
          +0tu36G3s1hfLhtH6ibKiRVErW5j2XixaQcpH0WQ+T6wDY5d1u
          G/38QpDCjPhWH7HlKKw==</pubkey>
     </client>
</sshclients>
```

Administrators must select Processing Metadata as the method for extracting the resource requested. The administrator must select ssh-password-metadata from the list of available metadata.

Administrators must select Any Authenticated User for the Authorization method.

Here is the command sequence to create an example AAA Policy to use for an SFTP Server Front Side Protocol Handler that uses public key authentication:

```
>co
>aaapolicy example-cc
aaapolicy>extract-identity
aaapolicy/extract-identity>method metadata
aaapolicy/extract-identity>metadata ssh-password-metadata
aaapolicy/extract-identity>exit
aaapolicy>authenticate
aaapolicy/authenticate>method custom
aaapolicy/authenticate>custom-url store:///ssh-client-auth.xsl
aaapolicy/authenticate>exit
aaapolicy>extract-resource
aaapolicy/extract-resource>method metadata
aaapolicy/extract-resource>metadata ssh-password-metadata
aaapolicy/extract-resource>exit
aaapolicy>authorize
aaapolicy/authorize>method anyauthenticated
aaapolicy/authorize>exit
aaapolicy>exit
>write-mem
```

It is easier to create the correct configuration by using the WebGUI on a different DataPower instance and then copying the resulting saved configuration to the machine that is conformant to the evaluated configuration. See the section below entitled Creating and Migrating Configuration for more information.

4.10 Connections to Remote Servers

Administrators must configure actions to use Transport Layer Security to establish a secure channel to trusted remote entities, such as authentication servers, or authorization servers or configuration servers. In some cases, it is possible to identify an SSL Proxy Profile within the configuration of the action (such as in AAA connections to LDAP servers). In other cases, such as with a Results action, Administrators must create an appropriate User Agent configuration that will enforce the use of TLS on connections. A User Agent configuration matches outbound URLs to a particular SSL Proxy Profile. See the section entitled “SSL Connections” above for detailed information about how to create a conformant SSL Proxy Profile.

A User Agent is referenced by an XML Manager which is referenced by the service that includes the MultiStep Processing Policy running the action making outbound connections. Here is an example showing the CLI commands used to create a User Agent that is then added to the configuration of an XML Manager.

```
user-agent "default"
  summary "Default User Agent"
  ssl "*" "example-ssl-proxy"
exit

xmlmgr "default"
xsl cache size "default" "256"
xsl checksummed cache default
no tx-warn "default"
user-agent “default”
xml parser limits "default"
exit
```

4.11 Creating and Migrating Configuration

The evaluated configuration does not allow the use of the WebGUI. The evaluated configuration does allow administrators to use the WebGUI on a different machine to create the CLI commands needed to create the configuration desired on the conformant machine. The domain configuration files found in the config:/// directory of any application domain, including the default domain, contain CLI commands. Administrators can use these files to see the CLI commands needed to create a desired configuration.

Administrators can copy these configuration files onto a conformant machine provided administrators carefully review the commands in the file to be sure all are within the boundaries of the evaluated configuration. Administrators must use the SCP, SFTP, or HTTPS transport protocols to copy configuration files onto a conformant machine.

Note that because the WebGUI cannot be running on a conformant machine, use of the WebGUI Import Configuration panel is not allowed.

After the new configuration file has been put in place in the config:/// directory of an application domain, the administrator must restart the domain to make the configuration take affect. Here is an example of the CLI command sequence required:

```
[AcceptanceCriteria]# restart domain
Restarting 'AcceptanceCriteria' will affect all services configured
within the domain!
Do you want to continue? [y/n]:y
Domain 'AcceptanceCriteria' restarted.
[AcceptanceCriteria]#
```

Once a configuration has been moved onto the conformant machine, the administrator must verify that the running configuration of the device is conformant. Issue the following command:

```
[AcceptanceCriteria]# show running-config
```

See Appendix B for a list of conformant commands.

4.11.1 Error Handling

Note: Users and administrator must be aware that If the configuration contains a command NOT listed in the Appendix B, the user/administrator is no longer operating in the evaluated configuration.

5 Tasks for AppDomain Administrators

In this section, we use two related business scenarios to present tasks that AppDomain administrators need to do to bring the TOE into operation as an Application-Level Firewall or Gateway.

Example Enterprises (formerly known as SNL Enterprises) sells a line of floor waxes and a line of wood stains. They also sell a line of wax/stain combination products.

Example's marketing department wants to provide information about all three product lines to consumers but does not want to expose its network servers or even its network topology to the world at large.

Example also wants to interact with its supplier through a web services-based tracking application. Example's logistics department wants only machines in the supplier's own networks to be able to access the tracking application.

Example's marketing department wants to provide information about its floor wax, wood stain, and "combo" product lines to consumers but does not want to expose its network servers or even its network topology to the world at large.

Example also wants to interact with its supplier through a web services-based tracking application. Example's logistics department wants only machines in the supplier's own networks to be able to access the tracking application.

Example's initial (privileged) administrator has created the following AppDomains and AppDomain administrators:

- ConsumerInfoServices with admin Peter
- SupplierLogistics with admin Holly

We will first look at the overall results that Peter and Holly are trying to achieve. As part of this, we will look at information flow from a high-level perspective. Following the two examples, we look at a firewall's components in more detail and show the specifics of how Peter and Holly create the firewalls they need.

Note: Most of the commands illustrated below start from "Global Configuration Mode". After an AppDomain administrator logs in, they must issue "configure terminal"

> **configure terminal**

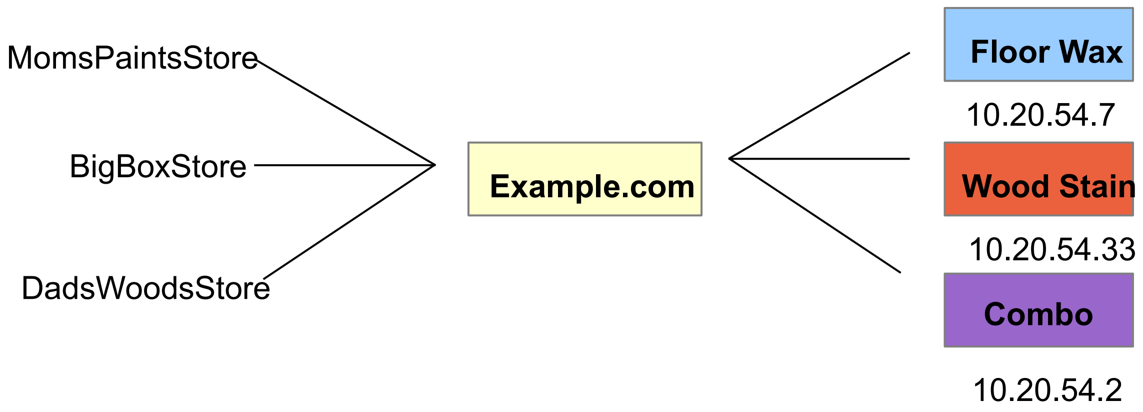
5.1 Consumer Information Services Firewall/Proxies

Example has three web applications — one for each product line — each located on its own server. These servers and their IP addresses are respectively:

www.fw.example.com	10.20.54.7
www.ws.example.com	10.20.54.33
www.combo.example.com	10.20.54.2

Example does not want its internal network topology exposed to the world at large. Using the TOE, AppDomain admin Peter can create a single entry point for all three web applications — www.toe.example.com — making it seem like all three applications are all hosted on the same machine.

The desired situation is shown in the diagram below. Consumers from anywhere in the world see the single host "toe.example.com" as the face of Example Enterprises. Each consumer gets their request routed by the TOE to the appropriate backend server (floor wax, wood stains, or combo).



The TOE includes three different services that can be used to meet these business requirements. These are:

XML Firewall (XI52 and XG45 only)
Multi-Protocol Gateway
Web Service Proxy

Many of the tasks are common to all three services to achieve the desired result. These are discussed below.

To realize this result, Peter created three services. (We will get to the details of service creation shortly.) Each serves as proxy for one of the backend servers (floor wax, wood stain, combo). Each service is associated with a different IP port. As you will see immediately below port use is reflected in the host portion of URLs that clients use.

The URLs that clients can connect to are:

For floor waxes:
http://www.toe.example.com/floor_wax/shiny.html
http://www.toe.example.com/floor_wax/matte.html

For wood stains:
http://www.toe.example.com/wood_stains/original.html
http://www.toe.example.com/wood_stains/dark_color.html

For combination floor wax/wood stain products:
http://www.toe.example.com/combo/shiny_original.html
http://www.toe.example.com/combo/shiny_dark_color.html

The TOE infrastructure internally routes each incoming message to its associated backend server (assuming the request passes other information flow policy rules).

Look at the service monitoring www.example.com to make this more concrete. This service listens on the customary port 80, thus requiring no explicit port number in the URL used by customers. It routes any customer request to the appropriate back-end server. So, for example, a request for floor wax would be routed to 10.20.54.33:8008 — the IP address and port of the floor wax server. Unknown to the customer, the request URL is modified to fit the request.

As a specific example**, http://www.toe.example.com/floor_wax/shiny.html** becomes **http://10.20.54.7:8008/shiny.html**

The other services work similarly, substituting the appropriate IP address for their server, and simplifying the file portion of the URL.

5.2 Business-to-Business Scenario

Example Enterprises interacts with its supplier through a web services-based logistics application. HTTP messages are used to carry SOAP/xml content between Example and its supplier. The logistics service is hosted on www.logistics.example.com.

www.logistics.example.com	10.20.54.23

Example wants only machines in the supplier's own networks to be able to access the logistics application. Using the TOE, AppDomain admin Holly can create an entry point that allows only traffic that (modulo IP spoofing) originates from the supplier.

To achieve this result, Holly creates a service that permits and denies traffic based on IP address. It does this through "allow" and "deny" rules that specify either single addresses or **address ranges.** Address ranges are specified through the common CIDR notation.

The supplier's networks are 10.52.43.0 and 10.98.22.0. Holly creates the following rules:

```
allow 10.52.43.0/24
allow 10.88.22.0/24
```

These rules specify that clients contacting the TOE from ranges 10.52.43.0 — 10.52.43.255 and 10.88.22.0 — 10.88.22.255 are permitted to connect to the TOE. The TOE will refuse a TCP connection with clients with addresses not in these allowed ranges.

What about routing?
The TOE still serves to mask the actual logistics server from the outside world.

5.3 Functional Components of a Service

Now that we've seen some examples of service use in a business context, we will look at the functional components of the service. This is essential background to service creation.

A service definition consists of the following:

1. An IP address and port number for HTTP clients to use to connect to the service. This is referred to as the "local address".

Note that: www.toe.example.com has the IP address of 10.99.8.8

For SupplierLogistics there is one service. Its local address and port is 10.99.8.8 220

2. The IP address and port number of the enterprise server that the service will send (allowed) client traffic to. This is referred to as the "remote address".

   Floor wax server: 10.20.54.7   8008
   Wood stains server: 10.20.54.33   8008
   Combo server: 10.20.54.2   8008
   Logistics server: 10.20.54.23 5555

3. An ACL (access control list) that contains the allow/deny rules that specify which client addresses can connect to the service and which are blocked. **All services must reference an ACL that blocks HTTP requests from internal (backend) which arrive at a firewall's (external) local-address[2].** (We provide details on creation of ACL objects shortly.)

4. A set of condition/action rules that can operate on the URL, the HTTP header, the content of a message or any combination. The collection of condition/action rules is referred to as a "stylepolicy".

The other required elements are the service name and the application domain that the service is part of. (Domain is set automatically by the TOE however.) Here is a pictorial representation of the firewall definition:

[2] Rejection of requests from internal addresses at the external firewall interface is a Common Criteria requirement.

XMLFIREWALL

Name: *string*
Domain: *string*
LocalAddress: *IPPort*
RemoteAddress:*IPPort*

ACL: *ACL object name*

Stylesheet-Policy: *policy object name*

Note that the firewall definition consists of both direct values (for name, domain, local-address, and remote-address) and **references** to other named objects (ACL and stylepolicy).

Before a service can be created, the objects the service administrator wants to reference must already exist. We will now look at the creation of ACL objects and stylepolicy objects. Then, we will give the steps for service creation.

5.3.1 The ACL Object

An ACL object consists of a series of "allow" and "deny" properties. Each allow and deny property consists of either an IP address or an IP address with a network mask. An IP address with a network mask allows for specification of a range of addresses.

The ACL object works on a "first match" principle when deciding whether to allow or deny a particular client connection. Consequently, the order of allow and deny properties is vital when creating an ACL.

Note that if the client is not explicitly allowed, the connection is denied. Here is a pictorial form of the ACL definition:

ACL
Name: *string*
Domain: *string*

Allow: ip-addr1
Allow: ip-addr2/mask
Deny: any

ACLs, like firewalls, each have a name and a domain that they are associated with. The above diagram shows the use of the "any" keyword. "Any" means "all". In the case of the above ACL, "deny any" is the default behavior (but we wanted to show the syntax).

In the case of the logistics proxy, Holly creates the ACL as follows:

> **acl** logistics_client_IPcontrols

where "logistics_client_IPcontrols" is the name Holly is giving to this ACL.

The acl command invokes ACL Configuration mode.

Now Holly sets the properties she wants.

> **allow** 10.52.43.0/24

> **allow** 10.88.22.0/24

Holly needs to block internal (backend) addresses from connecting to the service, but because the default behavior of the ACL is to deny all addresses which are not allowed, no explicit entry is required.

Lastly, Holly completes ACL creation by issuing the exit command:

> **exit**

Here is a pictorial form of the ACL Holly just created:

<u>ACL</u>
Name: logistics_client_IPcontrols
Domain: SupplierLogistics

Allow: 10.52.43.0/24
Allow: 10.88.22.0.24

Note that Holly did not explicitly set the domain name that is part of the ACL. The TOE does that automatically based on the domain that the administrator is in during ACL creation. The administrator does not have the ability to change the domain.

The logistics_client_IPcontrols ACL is now ready to be referenced by a service.

Peter must create an ACL that will be referenced by each service even though he is not specifically interested in blocking requests by client address. He must create an ACL to block (backend) addresses from connecting to the service (external) local-address. Here are the steps he must do:

```
> acl blockInternalAddresses

> deny 10.20.54.0/24

> allow any

> exit
```

This set of entries has the effect of blocking requests that appear to arise from the internal network but allowing all requests with external addresses.

Peter can reference this single ACL in each of his services.

5.3.2 The Stylepolicy Object

A stylepolicy object contains set of condition/action rules that can operate on either the URL, the HTTP header or the content of the message. The condition/action rules also allow/deny messages, and can also rewrite URLs, HTTP headers, or the content. We will look at how the stylepolicy object lets Peter accomplish his goals;

Recall that Peter, the AppDomain admin for the ConsumerInfoServices domain, wants to allow the following URLs related to floor waxes:

http://www.toe.example.com/floor_wax/shiny.html
http://www.toe.example.com/floor_wax/matte.html

Peter wants to allow client requests with these URLs to get to the TOE; he also wants to make his rules extensible however, so that when Example adds new types of floor waxes to its product line the information flow rules don't have to change. His actual *condition* then is that allowable URLs have the following form:

http://www.toe.example.com/floor_wax/*.html

Recall, that Peter also wants to change the URL before the request gets sent back to the server. Specifically, he wants to eliminate the "floor_wax" portion of the URL. So,

http://www.toe.example.com/floor_waxes/(.*).html is to become

http://10.20.54.33:8008/$1.html

where 10.20.54.33 is the address of the floor wax server. This is the *action* that Peter wants.

For a specific example**, http://www.toe.example.com/floor_wax/shiny.html** is to be changed to **http://10.20.54.33:8008/shiny.html**

Peter has another condition/action rule: If a client request has a non-allowed URL, he wants the service to reject that request rather than send it on to the floor wax server.

Peter has now conceptually defined both the *conditions* and the *actions* he wants for his floor wax service. Now look at a pictorial form of the stylesheet policy object:

Stylesheet Policy Object

Name: *string*
Domain: *string*

match *conditions-object1 actions-object1*

match *conditions-object2 actions-object2*

The "match" property allows for the specification of one or more conditions object and an actions object pairs.

That is, the conditions and actions are each **referenced** rather than being specified in-line.

Therefore, before we give the specifics of stylepolicy creation, we need to look at the conditions object (called "matching objects") and actions objects (called "rule" objects or "rule action" objects).

5.3.2.1 Matching Objects

The *matching-object* consists of a set of the following properties:

- "urlmatch" properties each of which specifies a URL or URL regular expression pattern that constitute a match
- "httpmatch" properties each of which specifies an HTTP header field and value. An HTTP message containing the header field and value specified in an httpmatch property is considered a match.

Here is a pictorial form of the *matching-object*:

Matching Object

Name: *string*
Domain: *string*

urlmatch: *url1*
urlmatch: *url*
httpmatch: *field value*

Peter only needs URL matches. He creates the matching object for his conditions with the following command:

```
> matching floor_wax_condition
```

This invokes the Matching Rule Configuration Mode, where floor_wax_condition is the name of the matching object Peter will create.

Because Peter wants to be very precise with his matches, he first sets the match mode to use PCRE rather than the simpler URL match mode.

```
> match-with-pcre
```

Now Peter creates the match expression with the following command:

```
> urlmatch (?i)^/floor_wax/[a-z]*$
```

Here '(?i)' causes the match to ignore alphabetic case. The '^' indicates the URI to match must begin with a forward slash followed by the letters floor_wax/. The expression '[a-z]' is the single character wildcard. It matches one occurrence of any single character in the alphabet between a and z inclusive. '*' is the string wildcard. It matches 0 or more occurrences of any character also in the range given. Finally, the '$' indicates that the URI must end here.

This expression allows the desired input of "/floor_wax/shiny" or "/FLOOR_WAX/SHINY" but not "/floor_wax/9999" or "/floor_wax/shiny/return all".

Lastly, Peter invokes exit to complete the matching object creation.

```
> exit
```

Peter needs three different matching objects to handle the three different kinds of requests the service must handle. He creates another match object with the following commands:

```
matching woodstain_condition
match-with-pcre
urlmatch (?i)^/wood_stain/[a-z]*$
exit
```

Lastly, he needs a matching object for combo requests. He issues these commands:

```
matching combo_condition
match-with-pcre
urlmatch (?i)^/combo/[a-z]*$
exit
```

Now, Peter will go on to create the rule action objects.

5.3.2.2 Rule Action Objects

The *rule action object* represents any one of the many actions available within the TOE. Refer to IBM Knowledge Center for complete information about what actions are available and how to use them.

For Peter's purpose, he needs an action to rewrite the URL given by clients to a URL that is understood by the back end server. He also needs to dynamically set the IP address and port of the back end server. Peter needs to use three different actions to accomplish this.

In order to rewrite the URL, Peter needs a Rewrite Action. A Rewrite Action uses a URL Rewrite Policy. First, he creates a URL Rewrite Policy object.

URL Rewrite Policy Object
The URL Rewrite Policy object consists of a series of properties directing transformations of a request's URL or HTTP header:

Here is a pictorial form of the URL rewrite object:

URL RewriteObject

Absolute-rewrite *input URL output-URL*

Header-rewrite *field regExp inputReplacefss*

Peter creates the url rewrite object he needs as follows

> **urlrewrite** ExampleRewrite

This invokes URL Rewrite Policy Configuration Mode. Now he specifies the rewrite rule. He only needs to rewrite the URL itself and not any HTTP header files. The only rewrite rule for the TOE is

> **absolute-rewrite** *input-URL-pattern output-URL*

Peter's rule is as follows:

> absolute-rewrite "(.*)/floor_wax/(.*)" "$1/$2" "" off on off

The first argument is the input URL pattern and the second argument specifies the output URL. Note that the output URL references the pattern portion of the input.

Note also that the regular expressions used in the URL rewrite object are slightly different than the regular expressions used in the matching rule object. Here '.' is equivalent to the use of the single character wildcard '?' employed in matching rule objects.

The last four parameters for the command set defaults. Peter obtained the correct values for these arguments by looking at the CLI **urlrewrite** command created on a test machine.

Finally, Peter exits out of the configuration mode.

> **exit**

Now that Peter has created the URL Rewrite Policy object, he can create the action. He issues the following commands:

```
action "example_rewrite"
  reset
  type rewrite
  urlrewrite-policy ExampleRewrite
exit
```

In order to dynamically set the IP address and port of the back end server, Peter needs to use actions. First, he sets a variable to the necessary values. He issues the following commands:

```
action example_setvar_fw
  reset
  type setvar
  input INPUT
  variable var://context/mine/route
  value http://10.20.54.7:8008
exit
```

Since Peter has three possible destinations, he must create a variable for each. He issues the following commands:

```
action example_setvar_ws
  reset
  type setvar
  input INPUT
  variable var://context/mine/route
  value http://10.20.54.33:8008
exit

action example_setvar_combo
  reset
  type setvar
  input INPUT
  variable var://context/mine/route
  value http://10.20.54.2:8008
exit
```

Now he can use this variable to set the route for the message. He issues the following commands:

```
action example_route-set
  reset
  type route-set
  destination var://context/mine/route
exit
```

Finally, he needs an action that connects the client request input to the service back end output. For this he uses the following commands:

```
action "example_results"
  reset
  type results
  input INPUT
  output-type default
exit
```

Now that the components of a rule are ready to use, Peter can create his processing rules. He needs three rules.

First, he creates the rule to handle floor wax requests. He issues the following commands:

```
rule example-rule-floor-wax
  reset
  type request-rule
    action example-rewrite
    action example_setvar_fw
    action example_route-set
    action example_results
exit
```

Now he creates the rule to handle wood stain requests:

```
rule example-rule-wood-stain
  reset
  type request-rule
    action example-rewrite
    action example_setvar_ws
    action example_route-set
    action example_results
exit
```

Here is the rule to handle combo requests:

```
rule example-rule-combo
  reset
  type request-rule
    action example-rewrite
    action example_setvar_combo
    action example_route-set
    action example_results
exit
```

Now Peter has both the matching objects that specifies conditions and the rule actions object that specifies actions. But he only has these for incoming HTTP requests. He will be able to use "built-in" matching and rule actions objects that will apply to the HTTP responses from the backend server. We show this below.

Peter can now create the stylepolicy object. Recall that the form of the "match" property is:

match *conditionsObject ruleObject*

Peter issues the following commands:

```
stylepolicy  ConsumerInfoPolicy
match   floor_wax_condition    example-rule-floor-wax
match   wood_stain_condition    example-rule-wood-stain
match   combo_condition    example-rule-combo
match  ALL  ACCEPT-RESPONSE
exit
```

Note: For the case where there is more than one "match" statement, the TOE selects the first one that matches the incoming message (be it a request or response). Thus **the order of match statements matters.** As with ACLs, it is important for the administrator to place the match statements in the order of the most specific matching condition to the least specific. In particular, any "catch-all" statement, must go last, otherwise it will block the selection of match statements with more specific matching conditions.

5.3.2.3 More about the Built-in Objects

The TOE contains the following built-ins:

Matching object: ALL

Rule Actions Objects: ACCEPT, ACCEPT-REQUEST, ACCEPT-RESPONSE, REJECT, REJECT-REQUEST, REJECT-RESPONSE

These objects operate as their names indicate. Matching object ALL matches all messages (requests and responses since matching objects do not specify direction). The Rule actions objects specify "direction" with ACCEPT and REJECT apply to both requests and responses.

You can specify matching object ALL with any rule action object you choose. Likewise you can pair a built-in rule action object with any matching object you choose.

The built-ins rule action objects allow you to pass-thru or block messages selected by the matching object you specify.

5.4 Creating the Example Services

To illustrate the use of each of the services, a different service will be built for each of the back end services. Each type of service (XML Firewall. Multi-Protocol Gateway and Web Service Proxy) will use some or all of the objects already built.
Peter now has all he needs to create the floorwax firewall proxy. Note that Peter does not need to specify an ACL to achieve his desired result. Holly has the ACL she needs but has not yet created a stylepolicy object. We will complete Peter's firewall, and get back to Holly's scenario shortly.

To create a firewall/proxy for the floor wax server, Peter does the following:

> xmlfirewall example-cc

This invokes Firewall Service Configuration Mode. Note that Peter is not using an XB62 appliance. Peter now specifies the local address, the remote address, and the stylepolicy object.

The only allowed form of the local address property in the TOE is:

local-address host   port

Peter specifies:

> local-address 10.99.8.8 80

Because the service dynamically determines the back end, the remote address command is as follows:

> remote-address %dynamic%

Now Peter specifies the stylepolicy object:

> stylesheet-policy ConsumerInfoPolicy

And the ACL:

> **acl blockInternalAddresses**

Peter must add the following commands to the firewall definition:

> **request-type xml**

This directive tells the firewall to apply automatic xml well-formedness checks and reject any request that is not well-formed xml.

> response-type passthru

This directive tells the firewall to pass the response from the server through to the client.

Peter has completed the firewall definition. He need only issue the exit command.

> exit

At this point, the TOE checks the validity of the firewall definition. If the firewall is valid, the TOE immediately instantiates the firewall at the address and port specified by the local-address property.

When Peter is happy that the firewall is working correctly, he should issue "**write mem**" command to save the configuration he has created to the file system. If he does not do so, his object definitions will be lost on system shutdown.

5.5 Secure Sockets Layer Communications

Some of the consumers of Example's services prefer to use SSL-enabled transports to place orders with Example. This is particularly good security policy for servers that handle sensitive payment information, such as credit card numbers. Peter must create an SSL Proxy Profile object first before he can complete the services.

At the heart of any SSL-enabled communication is a set of cryptographic keys. These keys are asymmetric, and come in pairs, consisting of a public certificate and a private key. To complete his configuration, Peter must first create the key objects.

Peter must have his public/private key pair available on a local system before he begins this task.

To begin, Peter copies the necessary keys to the appliance. Using the CLI, he issues these commands:

```
>copy scp://secureserver/datapower/SSLPubkey.pem cert:///SSLPubKey.pem

>copy scp://secureserver/datapower/SSLPrivkey.pem cert:///SSLPrivKey.pem
```

These two commands transfer the key files from a secure server to the encrypted storage area of the appliance. Once placed in the cert: directory, the keys cannot be moved or copied.

Now, Peter enters crypto mode:

```
> crypto
```

He then creates the two key objects he needs:

```
crypto>key ExSSLPrivKey SSLPrivKey.pem
```

where ExSSLPrivKey is the name of the key object being created and exampleprivkey.pem is the actual key file now stored in the encrypted cert: directory of the appliance.

```
crypto>certificate ExSSLPubKey SSLPubKey.pem
```

where ExSSLPubKey is the name of the key object being created and examplepubkey.pem is the actual key file now stored in the encrypted cert: directory of the appliance.

Now, Peter ties the two keys together into an Identification Credential:

```
crypto>idcred ExSSLSvr ExSSLPrivKey ExSSLPubKeycrypto
```

Now, Peter creates the CryptoProfile needed, disabling the use of SSLv3, TLSv1 and TLSv1.1 and explicitly specifying a set of strong ciphers:

```
crypto
  profile "ExSSLSvr" "ExSSLSvr" option-string OpenSSL-default+Disable-
SSLv2+Disable-SSLv3+Disable-TLSv1+Disable-TLSv1d1 ciphers "AES256-
SHA256:AES128-SHA256:AES256-SHA:AES128-SHA:DES-CBC3-
SHA@STRENGTH"
exit

crypto>exit

crypto>write mem
```

Finally, in Global mode, Peter creates the SSL Proxy Profile that will be attached to the example service:

```
>sslproxy ExSSLProxy reverse ExSSLSvr

>write mem
```

5.6 Multi-Protocol Gateway Configuration

Peter creates a Multi-Protocol Gateway to protect the combination server, as some of the customers buying these products use an SSL-protected transport and some do not. A Multi-Protocol Gateway can accept both types of transport simultaneously. He must create Front Side Protocol Handlers for each case.

5.6.1 Front Side Protocol Handlers

Now, Peter needs to create two Front Side Handler objects, one for handling HTTP transport requests and one for handling HTTPS transport requests.

He begins by issuing the Global command:

>source-http HTTP130

This creates the object with the given name and enters source-http mode. Peter completes the configuration:

source-http>acl blockInternalAddresses

This assigns the ACL to the Handler.

source-http>local-address 10.9.8.8

This sets the IP address of the listener.

source-http>port 130

This sets the listening port to 130.

source-http>exit

source-http>write mem

Now, he creates the HTTPS handler:

>source-https HTTPS131

```
source-https>acl blockInternalAddresses

source-https>local-address 10.9.8.8

source-https>port 131

source-https>ssl  ExSSLProxy
```

This assigns the SSL Proxy Profile to the listening handler, thus allowing inbound SSL connections.

```
source-http>exit

source-http>write mem
```

5.6.2 Gateway Service Configuration

Peter can now create the Gateway service. He begins by issuing the Global Mode command:

```
>mpgw ComboProxy
```

which creates the object and gives it the specified name. The CLI is now in Multi-Protocol Gateway mode. He completes the configuration:

```
mpgw>front-protocol HTTP130

mpgw>front-protocol HTTPS131
```

These two commands assign the Front Side Protocol Handlers to the Gateway, allowing it to listen for requests on port 130 or 131.

```
> request-type xml
```

This directive tells the gateway to check all inbound messages for well-formed xml.

```
> response-type passthru
```

This directive tells the gateway to apply policy to the server's response header.

> stylepolicy ConsumerInfoPolicy

This command assigns the same processing policy to the Gateway as was used for the firewall.

mpgw>type dynamic-backend

This establishes that the Gateway dynamically determines the back end URL, to which requests are forwarded. Recall that the Processing Policy rewrote the URI.

mpgw>exit

mpgw>write mem

This completes the configuration of the Gateway.

5.7 Web Service Proxy Configuration

For the Logistics Proxy, Holly wants to publish a WSDL for use by Example's suppliers. By using a WSDL, Holly establishes a standards-based method for coordinating requests from suppliers with the capabilities of Example's server applications. For this reason, Holly chooses to use a Web Service Proxy to protect Example's resources.

Like Peter, Holly needs a Front Side Protocol Handler:

>source-http HTTP220

This creates the object with the given name and enters source-http mode. Holly completes the configuration:

source-http>acl logistics_client_IPcontrols

source-http>local-address 10.9.8.8

source-http>port 220

source-http>write mem

source-http>exit

In order to do that, Holly needs to create a new Endpoint Rewrite Policy.

>wsm-endpointrewrite Logistics

This creates the object and names it.

wsm-endpointrewrite>listener-rule ".*" "http" "10.9.8.8" "220" "/status" "HTTP220" "on" "soap-11" ""
This creates the front side of the proxy, which listens for requests from suppliers using the HTTP220 Front Side Protocol Handler, and requires all requests to conform to the SOAP 1.1 specification.

wsm-endpointrewrite>backend-rule ".*" "http" "10.20.54.23" "5555" "/logistics" "" "" ""

This identifies the back end server to which requests are sent, including rewriting the URI.

Holly also needs to place the WSDL file on the appliance for use by the Proxy:

>copy http://files.example.com/wsdls/logistics.wsdl **local:///logistics.wsdl**

Now, it is necessary to build a stylepolicy that will pass requests through to the back end. Because Holly does not want to perform any operations on the incoming request or the outbound response, Holly can use a simplified version of the stylepolicy. She begins by creating the object from Global mode:

> **matching all**

This invokes the Matching Rule Configuration Mode, where logistics is the name of the matching object Holly will create.

> **urlmatch ***

Here '*' is the string wildcard. It matches 0 or more occurrence of any character.

Lastly, Holly invokes exit to complete the matching object creation.

> exit

Now, Holly will go on to create the rule actions object.

>action Results-all

This creates the action object and names it.

action>type results

This sets the action type to a Results action, which sends content where directed.

action>input "INPUT"

action>output "OUTPUT"

These two lines indicate that the contents of the INPUT context, containing the original request payload, are moved directly to the OUTPUT context, which is what is sent to the designated back end, without changing the content.

action>exit

Holly can now create an all-purpose rule for handling requests:

>wsm-rule logistics_rule_0

wsm-rule>action "Results-all"

wsm-rule>exit

Holly is now ready to create her stylepolicy.

>wsm-stylepolicy Logistics

This creates the policy object and names it.

wsm-stylepolicy>match `"fragmentid" "" "all" "logistics_rule_0" "" "http://example.com/logistics#dp.wsdlName(logistics.wsdl)"`

The second argument, "fragmentid" is a special keyword that indicates all content that conforms to the specifications of the WSDL. The last argument indicates that the rule is applied at the WSDL level of the Proxy hierarchy. Thus, all messages that match operations in the WSDL will pass correctly, and all others are rejected by default.

wsm-stylepolicy>exit

Recall that Holly created an ACL to control which clients can connect to the logistics proxy. She does not need to transform the URLs of the requests or change the HTTP header values.

Holly can now create the Web Service Proxy:

>wsgw Logistics

This creates the Proxy object and names it.

wsgw>endpoint-rewrite-policy Logistics

This assigns the endpoint policy, establishing the front side handler and back end destination.

wsgw>stylepolicy Logistics

This attaches the simple pass-thru policy to the Proxy.

wsgw> wsdl "local:///logistics.wsdl" "logistics.wsdl"

This identifies the WSDL used to generate the Proxy.

wsgw>exit

This exits the mode and returns to Global mode.

>write mem

This saves her definitions in the AppDomain's configuration file.

5.8 Modifying Configuration

The configurations demonstrated here represent minimal configurations providing significant protection. All three of the services demonstrated are capable of much more functionality, such as data transformation, dynamic routing, digital signing and signature verification, and so on. The full range of abilities are documented in the IBM Knowledge Center that accompanies this document. Information about creating conformant cryptographic configurations is given in Section 4 of this document.

For a list of commands that are allowed or not allowed to establish conformance with the Common Criteria specifications, see Appendix B of this document.

5.9 Updating Configuration

Note: If you make changes to an object that is part of a firewall or gateway definition, the new configuration takes place "immediately" — **however any connections that are in existence retain the settings they had when the connection was made.** For example, if you change the "remote-address", existing connections will continue to use the remote-address they started with.

If you want to ensure that all HTTP requests (and responses) conform to the new settings, you should:

- Save your configuration with "write mem" (if you have not already done so)
- Get a privileged admin to shutdown and restart the TOE with the "shutdown" command.

This will have the effect of terminating any existing connections through the appliance (for all domains).

6 Other Administrator Concepts and Tasks

6.1 Saving the In-Memory Configuration

When you make changes to the TOE, these are stored in working memory. Note that changes *always* are associated with a domain. For privileged operations, the domain is the "default domain".

If you want the changes in your domain to persist across reboots, you must save the changes with the **write memory** command. This command saves the changes made to the current domain to a domain-specific configuration file. If you do not do this, your changes will be lost. To save changes, you issue the following command:

> **write mem**

The TOE will ask you if you wish to overwrite the existing configuration file.

Overwrite existing <domain-specific config file>?

Type 'y' or 'yes'.

Note that when you do a "write mem" only the changes in the current domain get saved. That is, each domain has its own startup configuration file, with the system startup configuration file being that of the default domain. When the TOE boots, it executes the system startup configuration file, and then the AppDomain configuration files.

The implication of this structure is that if you are the AppDomain admin for more than one domain and have made changes in more than one domain, you must issue "write mem" in each domain that has undergone a change (e.g. a change in a matching rule) in order for the changes to persist across reboots.
Designating the System Startup File (Privileged Admins only)

By default, the system startup configuration file is **config:///autoconfig.cfg** in the default domain.

You, a privileged administrator, can choose to change the file that the TOE executes on boot with the **boot config** command. The reason to do this is as follows:

For example, you have a working system configuration but you wish to experiment — and possibly save the experimental configuration. The **write mem** command issued in the default domain always writes to config:///autoconfig.cfg . Without action on your part, this will overwrite your working default domain system configuration. To keep your working system configuration as the system startup configuration do the following:

1. Copy config://autoconfig.cfg to another file e.g.

> **copy** config://autoconfig.cfg config://stable.cfg

2. Designate the copied file as the system startup configuration file. Issue

> **flash**

This enters Flash Configuration Mode. Now issue

> **boot config** config://stable.cfg

This designates config://stable.cfg as the system startup configuration file.

Now exit out of the configuration mode. Issue

> **exit**

3. There is a final step you must take. Issue

> **no save-config overwrite**

This command tells the TOE to keep the file designated in the **boot config** command as the startup configuration. If you did not issue this command, your next use of **write mem** would not only save your running configuration in config:///autoconfig.cfg but also designate that file as the system's startup configuration file.

Note that if you wish to designate your running configuration as the startup configuration file again, you should issue

> **save-config overwrite**

```
> write mem
```

This will save your running configuration to config:///autoconfig.cfg and designate it as the system startup configuration file. Further uses of write mem will overwrite the file; the TOE will use whatever you have saved with write mem as the system startup configuration file.

6.2 Checking the Audit Log and System Log

Administrators (both privileged and application administrators) have the ability to check the audit log. To check the audit log, Administrators issue the following commands from any domain:

```
> configure terminal
> show audit-log
```

The TOE will provide a paginated listing of the audit log. The show audit-log command allows sorting by name, date, time, and address. Administrators can also search the log with the **show audit-search** command.

Guideline: Administrators should check the audit log at least once a day for signs of unusual activity or access attempts. Administrators should also check the default system log for Critical and Emergency errors. Guidance on handling Critical and Emergency error is given below in "Security-Relevant Events and Administrator Actions".

To check the system log, the administrator issues

```
> show log
```

The TOE will provide a listing of the system log.

Note: Never delete the system log (default-log). This object records critical and emergency errors

6.3 Audit Log Management and Archiving

In the evaluated configuration, only privileged administrators can manage the audit log.

An administrator should copy the oldest backup audit log file(s) to a secure machine located in the enterprise. This machine should itself be backed up regularly through the enterprise's backup procedures.

To copy the oldest rolled over audit file to a secure machine, do the following:

```
> copy audit:///audit-log.3  scp://USER@ipAddr/some/dir/backup-audit-<date>.txt
```

where "USER" is a legal username that you "own" at the host machine specified by "ipAddr".

You will be prompted to supply the password associated with the username USER.

You can of course specify whatever directory path and file name you choose. The above is an example that suggests you tag the backed up file with a date.

After archiving audit:///audit-log.1, you can remove it (if you wish) by issuing the privileged command audit delete-backup:

```
> audit delete-backup 1
```

Only the privileged administrator of the default domain can issue this command. See Section 1.3.6 Configure the Audit Log for more information about the audit log.

6.4 Log Management and Archiving (Privileged and AppDomain)

In addition to the audit file, the TOE has the notion of "log" files. The contents of audit and log entries are similar. In fact, every audit event is also recorded in the default; however, unlike audit entries, you can filter what events are sent to log files that are in your domain. Also, and importantly, log files are not maintained in persistent storage but rather in temporary memory that does not persist if you "shutdown reboot" the TOE. ("shutdown reload" maintains the log files).

You have two ways to filter system generated log notices: By "class" and by "priority". The list of event classes can be found by issuing the following command:

> **show logging event**

The event classes most relevant to the TOE are:

webgui : Web-Mgmt interface
auth : Authentication events
mgmt : Management events (e.g. configuration errors)
network : Network events
ssl : SSL events
system : System events
user : User events (e.g. resetting disabled accounts)
xmlfirewall : XML-Firewall events
all: All classes of events

You can find the list of logging priorities by issuing the following command:

> **show logging priority**

The priorities are:

emerg : Emergency: system is unusable
alert : Alert: action must be taken immediately
critic : Critical: critical condition
error : Error: error condition
warn : Warning: warning condition
notice : Notice: normal but significant condition
info : Information: informational messages
debug : Debug: debug-level messages

6.4.1 Pre-existing Logs and New logs

Each domain has its own "default log" that automatically stores the following events:

- "all classes" where the event is at the priority "error". In other words, all errors in objects in your domain are automatically logged.

- management class events that are at the priority of "notice" and higher.

The name of this default log is **logtemp:///default-log**. Note that each domain (including the "default" domain) has its own logtemp: directory.

You need take no action to get the logging of errors and management notices. The log is created for you.

New Logs
You can choose to create other logs in your domain that you can tailor to capture the event classes and priorities of your choosing. You create a log by issuing the following command:

> **logging target** *log-name*

This evokes Logging Configuration Mode.

You can then tailor your new log by issuing the following command:

```
> event   event_ class1  event_priority1

> event  event_class1  event_ priority2

> type file
> …
> exit
```

This new log will now capture the event classes/priorities that you chose. Note that your default log will still capture "all" errors and management notices — even if you are capturing errors and management notices in your new log. That is, the logging event model is "publish-subscribe". Each log you create will get all the events that it has been configured for.

Modifying the Configuration of Existing Logs
You can modify the settings of your logs (including the "default" log) by invoking Logging Configuration Mode passing in the name of your log. You need only provide the leaf name, for example, default-log, rather than the full pathname.

6.4.2 Log Sizes, "Rollover", Archiving and Deletion

You can control the size of your log files. To do so, you issue the following commands:

> **logging target** *log-name*

> **size** *kilobytes*

> **exit**

When your log file reaches the size in kilobytes you specified, it "rolls over". That is, the TOE renames it to <log name>.1 and begins a new log named <log name>. In the case of the default log for your domain, the rolled over log becomes logtemp:///default-log.1. When the new log now reaches its size limit, both it and the previously rolled over log are renamed. Using the default-log as an example, here is what happens:

default-log.1 becomes default-log.2
default-log becomes default-log.1
The new log is default-log

The next time the log "fills up", the oldest generation of the log gets automatically deleted. That is the current <log name>.2 disappears.

To save the rolled-over log files, you can copy them to a secure machine in your enterprise (that is itself backed up).

We recommend that AppDomain administrators regularly backup their rolled over log files, however it is a **must** that a **privileged admin backup the rolled-over default domain system logs each day.** Reason: The TOE logs security relevant events in the default domain system log. The file name is logtemp:///default-log.* .

To copy the log file to a secure machine, issue the following command where "USER" is a legal username that you "own" at the host machine specified by "ipAddr":

> **copy** <log name>.2 scp://USER@ipAddr/some/logdir/log_name-<date>.txt

You will be prompted to supply the password associated with the username USER.

You can, of course, specify whatever directory path and file name you choose. The above is an example that suggests you tag the backed up file with a date.

While the TOE will delete the ".2" generation of log file automatically, we suggest deleting the files you have archived immediately with the "delete" command. This will free up file system space. You issue one or more of the following command:

> **delete** logtemp:///<log name>.2

> **delete** logtemp:///<log name>.1

6.5 Backup and Restore of Configuration Files and Other Files

6.5.1 Backup and Restore of Configuration Files

Default Domain Config File
The system startup configuration file can be designated (by a privileged admin) through the "boot config" command (see "Designating the System Startup File). In lieu of administrator designation, the file is config:///autoconfig.cfg. To extract the boot config files, a privileged administrator issues the following command:

> write startup-config *leaf-file*

This extracts the boot config to config:///*leaf-file* in the default domain. You can then copy this file off of the appliance for backup purpose the same way you copy log files.

> **copy** config:///leaf-file scp://USER@ipAddr/some/dir/backup-startupconfig-<date>.txt

Note that "some/dir/backup-startupconf-<date>.txt" will be place under **home directory** of USER, where USER is an authorized user of the remote machine. All component except the leaf name must already exist under the home directory of USER, otherwise the copy fails.

You can **restore** the file by reversing the process:

> **copy** scp://USER@ipAddr/someotherdir/backup-startupconfig.txt config:///restore.cfg

Note that this simply brings the file back to the TOE's file system. To designate this file as the boot config file you must issue the following commands:

> flash
> boot config restore.cfg

```
> exit
```

Finally, for the file to be used as the startup config, you must reload the TOE with the shutdown command. The shutdown command exists in a configuration mode "above" Global Configuration Mode called "Privileged Configuration Mode". To get into Configuration Mode you must first issue the "top" command. Then you can issue the shutdown command:

```
> top

> shutdown reload
```

AppDomain Config File
Each AppDomain has its own configuration file and its own configuration directory *config:*. The name of an AppDomain configuration file is config:///<AppDomain name>.cfg. That is, the configuration file for a domain consists of the domain's name plus the ".cfg" suffix.

Going back to the business scenarios, the config file for the ConsumerInfoServices domain is config:///ConsumerInfoServices.cfg

To back up the AppDomain's config file, issue the following commands:

> **switch domain** <domain name> :

> copy config:///<domain name>.cfg scp://USER@HOST/config/domainName-<date>.cfg

You can of course specify whatever directory path and file name you choose. The above are examples suggest that you tag the backed up file with a date.

To ***restore*** the file, simply reverse the process,

> copy scp://USER@HOST/config/domainName-<date>.cfg config:///<domain name>.cfg

The restored config does not take effect until you shutdown and reload (or reboot) the TOE. Also, if you issue a "write mem" from within the AppDomain, the config file will be overwritten.

6.5.2 Archiving Other Files

You might want to archive files that are in your domain. To do this first find the files by issuing the following command:

```
> dir local:
```

This gives a listing of the files within the current domain. Then, execute the copy command (as demonstrated above) for each file you wish to save.

6.6 Checking Object and Service Status after Reboot (Privileged Admin)

Any time the system boots or reboots, the administrator should observe the messages shown during the boot up sequence to verify that the random number generator was properly initialized. A serial line connection to the device is required to see these messages.

This is the output of the random number generator startup sequence:

```
 TPM 1.2 Version Info:
 Chip Version:        1.2.3.17
 Spec Level:          2
 Errata Revision:     2
 TPM Vendor ID:       IFX
 Vendor Specific data: 03110008 00
 TPM Version:         01010000
 Manufacturer Info:   49465800
```

If for any reason the initialization fails, the default system log will contain a CRITICAL error message.

After the system reboots from a failure, you must check for system errors.

To find system and "default domain" errors after reboot, you issue the following command:

```
> co
> show startup-errors
```

The TOE will provide a listing of errors in the system startup configuration. There should be no errors found. However, if there are errors shown, you must not operate the TOE as the TOE may not be properly running in the evaluated configuration.

Then, follow the instructions under "Saving Error Information".

6.7 As Needed Privileged Commands and Tasks

6.7.1 Re-enabling disabled accounts

The audit log will contain entries for each failed login attempt. If an admin cannot login, you should examine the audit log to see if there is, first of all, an entry stating that the admin is locked out. You should also try to determine whether the lock out is due to a simple failure to remember the password, or if there is evidence of a concerted attempt to break in. This evidence consists of multiple authorization failures, either with a single user name or with multiple possibly random user names. (In the former case, check with the user to see how many times they tried to log in before giving up.)

To re-enable an account that has been disabled due to too many login failures, you issue the following command:

> **reset username** *account-name*

For example,

> **reset username** Holly

You will be prompted to provide a password and to confirm that password.

Use this command after receiving a request from the disabled administrator, and verifying that the lockout is not the result of an intrusion attempt; examining the audit log will be helpful here.

6.7.2 Add New Privileged Administrators

To create a new privileged you do the following:

> username *new_admin_name*

> password *password_string*

You will be prompted to re-type the password as confirmation of your choice. Do so.

> access-level privileged

> exit

6.7.3 Deleting Privileged and AppDomain Administrators

Use the following command for deleting an administrator (either privileged or AppDomain):

> no username *admin_to_be_deleted*

6.7.4 Reinitializing the System

The “reinitialize” command under Flash Configuration Mode is only used in the event of a catastrophic failure, or before removing an appliance from the network for return or relocation. To reinitialize the system, issue the following commands:

> **flash**

> **reinitialize <leaf-name>**

where "leaf-name" is the name of a firmware image provided to you by IBM Support. This file must be in the "image:" directory.

> **exit**

6.7.5 Changing the Configuration of the Communications Ports

Interface configuration mode allows for the configuration of the physical network ports of the TOE.

To enter the configuration mode, the privileged administrator issues:
interface {
[ethernet 10 | eth10] |
[ethernet 11 | eth11] |
[ethernet 12 | eth12] |
[management 0 | mgt0]
}

where the parameter specifies the name of a physical network port.

Detailed documentation for the configuration mode is in the Interface Configuration Mode section of the command reference guide.

The TOE-allowed subcommands are
ip address
arp
ip default gateway
ip route

6.7.6 Network Configuration Changes

Network Setting Configuration Mode allows the privileged administrator to control network-related settings of the TOE such as ICMP message behavior, routing, and interface isolation.

To enter the configuration mode, the privileged administrator issues:
> network

The allowed TOE network subcommands are:
destination-routing
icmp-disable
tcp-retries
arp-interval

arp-retries
ecn-disable

See the command reference guide for information on the use of the subcommands.

The administrator issues the no form of this command to reset network settings to their default state:
> no network

6.7.7 Shutting Down the TOE

The **shutdown** command allows a privileged admin to fully halt, reboot, or reload the TOE.

For example, use “shutdown halt” prior to a planned power-down.

Issue the following command:

shutdown { reboot | reload | halt } [pause]

where reboot, reload, and halt are mutually exclusive keywords.

- reboot shuts down the operating system and restarts the TOE
- reload restarts the TOE
- halt shuts down the TOE but does no restart

- pause is an optional integer (within the range 0 through 65535) that specifies the number of seconds before the appliance begins the shutdown operation.

Note that a shutdown both preserves the counts of login failures and the locked-out (or valid) statuses of accounts.

Note also that shutdown can only be issued from Privileged Configuration Mode (and Flash Configuration Mode). A privileged admin can get to Privileged Configuration mode by issuing the command "top" from other configuration modes.

6.7.8 Starting Up the TOE

Startup of the TOE is by use of the physical toggle switch on the TOE's case.

6.7.9 Checking Memory and File System Usage

Memory

To prevent over-utilization of working memory, you should periodically check the state of the TOE memory usage. To do so, the you issue the following command:

> show memory

The TOE will provide a listing of memory utilization. For example

> show memory

```
total memory: 2017.2 MB
used memory: 528.7 MB (26.21%)
free memory: 1488.5 MB (73.79%)
Shared Memory: 0 kB (0.000000%)
Buffers: 2992 kB (0.2909197%)
```

If memory use increases, and appears that it will approach 100%, consider shedding some of the data-processing load on to a separate appliance. If you believe the load is not excessive, contact IBM Support to discuss a memory allocation defect.

File system

You can check the file system with the following commands. At the config prompt enter, "show filesystem" (note that this command is not in the printed documentation, but is in the online help):

> show filesystem

This will cause the TOE to display a listing of free and used space similar to the following list:

```
    free encrypted space: 525 Mbytes
    total encrypted space: 4103 Mbytes
    free unencrypted space: 525 Mbytes
    total unencrypted space: 4103 Mbytes
```

You can remove unneeded files with the CLI **delete** command. To find candidates for deletion in all domains, a privileged administrator switches to each domain, and lists the files using the **dir** command. You issue the following commands:

> **switch** domain *domainName*

> **dir** *directory*

Per-domain directories that are useful to look at are local:, config:, logtemp:, and temporary:.

The TOE will provide a listing of files.

Unneeded files can then be removed:

> **delete** temporary:foo.txt

Obvious candidates for deletion include rolled over log files (in the logtemp: directory). You should archive these files first though as discussed in "Backup Configuration and Other Files".

It is prudent to ask the AppDomain admin to clean up their own AppDomain, but if you need to take the action yourself, make sure to only delete administrator-created files and not system files. If you are not sure whether or not a file is a "system" file provided as part of the DataPower installation, you must contact IBM Support. Do not attempt "trial and error" deletion. Reason: Many of the system files are crucial for the operation of the appliance.

6.8 On-going AppDomain Tasks and Commands

6.8.1 Examine new service resource usage

After creating a new service (firewall or gateway), you should check both memory and your AppDomain's file system usage multiple times a day to see if the new service is consuming excessive system resources. After your new service is stable, you can check once a day.

Checking Memory

To check on working memory utilization you issue

```
> show memory
```

The TOE will provide a listing of memory utilization. For example:

```
> show memory

total memory: 2017.2 MB
used memory: 528.7 MB (26.21%)
free memory: 1488.5 MB (73.79%)
Shared Memory: 0 kB (0.000000%)
Buffers: 2992 kB (0.2909197%)
```

If memory use increases, and appears that it will approach 100%, notify a privileged administrator (such as the one who created your account).

Checking the AppDomain's File System

Use the **dir** command to list files and sizes in the AppDomain file system. For example:

```
> dir logtemp:
```

This will cause the TOE to display something akin to the following list:

```
default-log
default-log-xml
```

You can remove unneeded files with the **delete** command as discussed in section "Backing Up Configuration Files and Other Files". You should examine your local:, config:, logtemp:, and temporary: directories to look for candidates for deletion.

If you are the AppDomain administrator for another domain, you can get to that domain by the **switch domain** command.

> switch domain *AppDomainName*

You can only "switch" to a domain that you are an administrator for. Once you are in the domain, examine the per-domain directories listed immediately above for unneeded files.

6.8.2 Backup of Configuration and Log Files

You must backup your files on an ongoing basis.

See the sections "Log Management and Archiving", and "Backup Configuration Files and Other Files" for instructions.

7 Miscellaneous Information and Guidance

7.1 Other Communications-relevant Functionality

The TOE will not accept messages that do not conform to their protocol specification. Also, the TOE will not accept any data traffic on a port prior to an authorized administrator creating a firewall that accepts messages on that port. Finally, the TOE will also not accept messages that contain:

- Mismatches between the source address and the TOE interface on which the message arrives.
- Source addresses on a broadcast network
- Source addresses on the loopback network
- Messages that specify routing

These policies are "mandatory." Administrators have no discretion to change these aspects of the TOE's information flow handling.

7.2 Password Change Guidance

The TOE requires each administrator to change their password every ninety (90) days by "expiring" the password. Upon expiry of their password, the administrator will be prompted for a new password when they next login.

Administrators also have the option of changing their own password prior to expiration with the "user-password" command.

All administrators should follow good "hygiene" in creating and maintaining their passwords. Passwords should not relate to an administrator's personal information (such as, their name, birthdate, or other personal information) in any obvious way.

The TOE keeps a history list of your last five (5) passwords. You will not be able to change your password to one that is on the list.

Your password must be at least fourteen (14) characters in length and contain at least one non-alphanumeric character.

7.3 Security-Relevant Events and Administrator Actions

The TOE will log security-relevant system events in the default system log. Privileged admins can access this log via the **show log** command.

Each entry in the system log is annotated with a priority designator. "Critical" and "Emergency" messages require action on the part of an administrator. The following list contains the messages and administrator actions.

Configuration Issues

CRITICAL Domain configuration '<name>' invalid: <reason>
CRITICAL Error opening domain configuration '<name>'
CRITICAL Property <name**>** **has** a value beyond the supported length. It has been truncated.

These errors, while theoretically possible, should not occur. Contact IBM Support.

Resource-related errors
CRITICAL Watchdog system detects no activity for at least 45 seconds
CRITICAL Unable to initialize XML parser
CRITICAL Failed to construct XML tree: <name>
CRITICAL Memory and Sockets now available, re-enabling connections
CRITICAL Resources very low (mem=<percent**>** **ports**=<how many>), throttling connections
EMERGENCY Resource shortage has not recovered in %d <number**>** **seconds**, forcing restart.
EMERGENCY Free memory too low (<percent>), forcing restart
CRITICAL Failed to create attachment manifest: <reason>
CRITICAL User Agent: Could not open '<name>' - Port pool exhausted
CRITICAL Error creating temporary config script
CRITICAL Error writing to temporary config script
CRITICAL Error reading temporary config script
CRITICAL Cannot open script log file <name>.
CRITICAL get-log: Cannot open temp file
CRITICAL get-domains: Cannot open temp file

These errors tend to indicate temporary resource limitations (for example, low memory). The TOE will restart on its own if it is in a hung state, however you should examine the log (with a ***show log*** command at the command prompt) for transient loads and determine the cause of the load spike. If these errors continue, contact IBM Support.

CRITICAL Duplicate Address <IP address**> detected**. Conflict with host at <MAC address>

This conflict needs to be resolved by your organization's network management personnel. Contact them.

Licensing Errors
CRITICAL Licensing: license file failed signature validation: <reason>
CRITICAL Licensing: <reason>
CRITICAL Licensing: could not access the unsigned data
CRITICAL Licensing: this device's serial number is unknown
CRITICAL Licensing: failed to write out rootcert
EMERGENCY Failed to access DataPower license

These errors indicate that consistency checks related to licensing failed. Contact IBM Support. Note that some TOE features might not be available until the licensing issue is resolved.

Initialization Errors
EMERGENCY Errors initializing filesystem
CRITICAL System configuration script not found
CRITICAL System configuration generated errors
CRITICAL Default configuration script not found
CRITICAL Default configuration generated errors
CRITICAL <name>: error parsing generated XML: <reason>
CRITICAL get-log: Cannot parse temp file: <name>
CRITICAL: Common Criteria: Entropy is insufficient for pseudo-random number generation

These are all internal errors that theoretically should not occur. Contact IBM Support.

CRITICAL error creating <name>: <reason>
CRITICAL error writing to <name>
CRITICAL Failed to save configuration to <name>

CRITICAL Failed to change boot config to '<name>' - <reason>
CRITICAL Failed to save configuration to '<name>'
CRITICAL <name>: Cannot create temp file
CRITICAL <name>: Cannot open temp file

Make sure the flash portion of the file system is not full. To check the free space available, type the following as a privileged administrator (note that this command is not in the printed documentation, but is in the online help):

> show filesystem

This command will cause the TOE to display a listing of free and used space similar to the following list:

```
free encrypted space: 525 Mbytes
total encrypted space: 4103 Mbytes
free unencrypted space: 525 Mbytes
total unencrypted space: 4103 Mbytes
```

For guidance on removing unneeded files, see "Miscellaneous Guidance," below.

If your flash is not full, contact IBM Support.

Miscellaneous Issues and Errors

CRITICAL Shutdown scheduled in <number**> seconds**

Contact the administrator who scheduled the shutdown to validate the reason for the shutdown.

EMERGENCY User did not accept DataPower license

This message only occurs on refusal of the license terms. Contact your DataPower salesperson.

EMERGENCY User '<name>' is member of invalid group <name>
EMERGENCY Failure while reading password database
EMERGENCY Failed to commit password database
EMERGENCY Failed to commit password database: <reason>
EMERGENCY Failed to commit group database: <reason>

These errors indicate consistency issues on internal files. Contact IBM Support.

7.4 Handling Crashes (Privileged Admins)

In general, the TOE is "self-healing": if there is a crash, the TOE will restart automatically. The most likely reasons for a crash are either resource failure or a transient condition that was not handled properly.

If the TOE crashes repeatedly, it will enter "fail safe" mode. Repeated crashes are likely due to one or more configuration problems.

You can tell that you are in fail safe mode by the command prompt you will see after you login over the serial connection. (Network-based administration will be unavailable.) Here is the format of the prompt

fail safe >   .

When the TOE is in fail safe mode, it will not allow any data traffic. It will, however, let you do problem determination. A privileged administrator can and should examine the error report file, config files, the audit files, and log files.

For more information, see the discussion under "Saving Error Information".

After you have determined the problem and corrected it, you can bring the TOE out of fail safe mode by issuing the following command:

> **shutdown** restart.

This restarts the communications functions of the TOE.

Occasionally, a problem will occur (for example, a network interface becomes inoperable) that requires a reboot (a more drastic action than a restart). In this case, a privileged administrator must access the machine via the local serially-connected terminal and issue the command **shutdown reboot.** Alternatively, an administrator might need to power cycle the machine using the switch on the TOE's case.

7.5 "Help"

The **help** command provides online help for a specific command or displays a list of commands available in the current configuration mode if no command is provided. Documentation for the command is in the Common Commands section of the command reference. Note that help will only recognize commands that are available to the admin in the current configuration mode.

The administrator issues the following command where *commandName* is an optional command:

help [*commandName*]

Here are two examples:

```
> help xmlfirewall

xmlfirewall [name]
 Enters XML Firewall Configuration Mode and creates a
 named firewall.
 In the absence of the optional name argument, the system
 software generates a unique firewall name.
 For example:

 xmlfirewall FW-1
       enters XML Firewall Configuration Mode and creates the
       XML Firewall, FW-1


>  help clock
clock <yyyy-mm-dd | hh:mm:ss>
 Sets the date or time.
 For example:

 clock 2003-12-23
       sets the date to December 23, 2003
 clock 23:32:00
       sets the time to 11:32:00 PM.
```

7.6 Checking on use of Non-CC commands

The file logtemp:cli-log in the default domain contains a list of all commands that have been issued.

A privileged admin can access this file to check for use of non-CC commands by issuing the following commands:

```
> switch  domain default

> show file  logtemp:cli-log
```

Appendix A: Required Conditions for Network-Based Administration

The TOE allows network-based administration through the use of the CLI over SSH. (Note that telnet functionality is not part of the TOE.) By default, all network-based access is disabled, and only administration via the CLI over the serial port is permitted.

In order for the use of the CLI-over-SSH on an administrator station to be fully equivalent to the CLI via the directly connected terminal from a security perspective, there must be no way for a malicious person to install password monitoring software or appliances on the administrator station.

The following must apply:

- The network administrator stations must be on a private protected network that is not used for data traffic.
- The administrator stations are physically secure.
- The administrator stations must not be remotely controllable. Administration must require the physical presence of an authorized administrator.
- The administrator stations do not host public data.

On the TOE side, a privileged administrator should create an ACL for SSH (with "acl ssh") that explicitly allow only the IP addresses of the administrator stations.

If above constraints are fulfilled, then network-based administration is equivalent from a security perspective to serial port based administration.

Appendix B: CLI Commands in the Evaluated Configuration

In the evaluated configuration, the following commands can be used by administrators. Other commands available in the CLI pertain to functionality are not allowed in the evaluated configuration; their execution may put the TOE out of the evaluated configuration.. Refer to IBM Knowledge Center provided with this document for information about each of these commands.

- **aaapolicy**
- **account**
- **acl**
- **acl ssh**
- **action**
- **admin-state**
- **alias**
- **appliance-quiesce**
- **appliance-unquiesce**
- **audit delete-backup**
- **audit level**
- **audit-reserve**
- **audit-log-settings**
- **b2b-cap**
- **b2b-cpa-collaboration**
- **no b2b-cpa-collaboration**
- **b2bgw**
- **b2bp-archive-purge-now**
- **b2b-persistence**
- **b2b-ha-switch-primary**
- **b2b-initialize-secondary**
- **b2b-profile**
- **b2b-profile-group**
- **b2b-view-mgmt**
- **b2b-xpath-routing**

- **backup**
- **cache schema**
- **cache stylesheet**
- **cache wsdl**
- **cancel**
- **cpa-receiver-setting**
- **cpa-sender-settting**
- **clear aaa chae**
- **clear dns-cahe**
- **clear intrusion-detected**
- **clear ldap cache**
- **clear pdp cache**
- **clear rbm cache**
- **clear xls cache**
- **cli remote open**
- **client-known-host**
- **clock**
- **compact-flash**
- **compact-flash-initialize-filesystem**
- **compact-flash-repair-filesystem**
- **configure terminal**
- **conformancepolicy**
- **copy**
- **crl**
- **crypto**
- **crypto-export**
- **crypto-import**
- **decrypt**
- **delete**
- **detect-intrusion**
- **diagnostics**
- **dir**
- **disable vlan-sub-interface**
- **disconnect**
- **dns**

- **document-crypto-map**
- **documentache**
- **domain**
- **echo**
- **enable**
- **encrypt**
- **exec**
- **exit**
- **file-monitoring**
- **file-permissions**
- **flash**
- **ftp-quote-command-list**
- **help**
- **host-alias**
- **idcred**
- **import-execute**
- **import-package**
- **include-config**
- **input-conversion-map**
- **interface**
- **ip domain**
- **ip host**
- **ip name-server**
- **ipmi-Ian-channel**
- **ipmi-user**
- **json-settings**
- **key**
- **keygen**
- **known-host**
- **language**
- **ldap-search-parameters**
- **load-interval**
- **loadbalancer-group**
- **locate-device**
- **logging category**

- **logging event**
- **logging eventcode**
- **logging eventfilter**
- **logging ipfilter**
- **logging object**
- **logging target**
- **logging trigger**
- **login**
- **loglevel**
- **logsize**
- **matching**
- **max-login-failure**
- **mcfilters**
- **mcf-httpheader**
- **mcf-httpmethod**
- **mcf-httpurl**
- **message-matching**
- **message-type**
- **metadata**
- **mkdir**
- **monitor-action**
- **monitor-count**
- **monitor-duration**
- **move**
- **mpgw**
- **mpgw-error-action**
- **mpgw-error-handlingmtom**
- **network**
- **no aaapolicy**
- **no acl**
- **no action**
- **no alias**
- **no application-security-policy**
- **no b2b-cpa**
- **no b2b-cpa-collaboration**

- **no b2bgw**
- **no b2b-prifle**
- **no b2b-profile-group**
- **no b2b-xpath-routing**
- **no comforancepolicy**
- **no cpa-receiver-setting**
- **no cpa-sender-setting**
- **no deployment-policy**
- **no dns**
- **no document-crypto-map**
- **no deployment-policy-variables**
- **no domain**
- **no failure-notification**
- **no forms-login-policy**
- **no ftp-quote-command-list**
- **no fwcred**
- **no globallogipfilter**
- **no hostalias**
- **no httpheader**
- **no httpserv**
- **no idcred**
- **no import-package**
- **no input-conversion-map**
- **no ims**
- **no include-config**
- **no interface**
- **no ip domain**
- **no ip host-aliasno ip name-server**
- **no ipms-Ian-channel**
- **no ipmi-usre**
- **no jason-settings**
- **no kerberos-kdc**
- **no kerberos-keytap**
- **no key**
- **no known-host**

- **no ldap-connection-pool**
- **no ldap-search-parameters**
- **no loadbalancer-group**
- **no logging caterory**
- **no logging event**
- **no logging eventfilter**
- **no logging ipfilter**
- **no logging object**
- **no logging target**
- **no matching**
- **no mcfilters**
- **no mcf-httpheader**
- **no mcf-httpmethod**
- **no mcf-httpurl**
- **no message-matching**
- **no message-type**
- **no metadata**
- **no monitor-type**
- **no monitor-action**
- **no monitor-count**
- **no monitor-duration**
- **no mpgw**
- **no mpgw-error-action**
- **no mpgw-error-handling**
- **no mtom**
- **no nfs-client**
- **no nfs-dynamic-mounts**
- **no nfs static-mount**
- **no ntp**
- **no ntp-service**
- **no-oauth-supported-client**
- **no oauth-supported-client-group**
- **no odr**
- **no odr-connector-group**
- **no packet-capture-advanced**

- **no password-map**
- **no peer-group**
- **no policy-attachments**
- **no policy-parameters**
- **no profile**
- **no radius**
- **no rule**
- **no save-config overwrite**
- **no scc**
- **no search results**
- **no simple-rate-limiter**
- **no slm-action**
- **no slm-cred**
- **no slm-policy**
- **no slm-rsrc**
- **no slm-sched**
- **no smtp-server-connection**
- **no snmp**
- **no soap-disposition**
- **no source-as1**
- **no source-sa2**
- **no source-as3**
- **no source-ebms2**
- **no source-ftp-poller**
- **no source-ftp-server**
- **no source-http**
- **no source-https**
- **no source-ims-callout**
- **no source-imsconnect**
- **no source-raw**
- **no source-sftp-poller**
- **no source-ssh-server**
- **no source-stateful-tcp**
- **no source-tibems**
- **no sshclientprofile**

- **no sslforwarder**
- **no sskey**
- **no sslproxy**
- **no statistics**
- **no stylepolicy**
- **no stylesheet**
- **no tam**
- **no tcpproxy**
- **no throttle**
- **no tibems-server**
- **no tx-warn**
- **no uddi-registry**
- **no uddi-subscription**
- **no urlmap**
- **no urlrefresh**
- **no urlrewrite**
- **no user**
- **no user-agent**
- **no usergroup**
- **no valan-sub-interface**
- **no valcred**
- **no wcc-service**
- **no web-application-firwall**
- **no web-mgmt**
- **no webapp-error-handling**
- **no webapp-gnvc**
- **no webapp-request-profile**
- **no webapp-response-profile**
- **no webapp-session-management**
- **no wsgn**
- **no wsm-agent**
- **no wsm-endpointrewrite**
- **no wsm-rule**
- **no wsm-stylepolicy**
- **no xacml-pdp**

- **no xml validate**
- **no xmlfirewall**
- **no xml-manager**
- **no xml-mgmt**
- **no xpath-routing**
- **no xsl checksummed cache**
- **no xslconfig**
- **no xslcoproc**
- **no xslproxy**
- **no xslrefresh**
- **no zos-nss**
- **no certificate**
- **no crl**
- **no fwcred**
- **no idcred**
- **no kerberos-kdc**
- **no kerberos-keytab**
- **ntp**
- **ntp-service**
- **packet-capture-advanced**
- **password-map**
- **pwd-aging**
- **pwd-digit**
- **pwd-history**
- **pwd-max-history**
- **pwd-max-age**
- **pwd-max-history**
- **pwd-minium-length**
- **pwd-mixed-case**
- **peer-group**
- **ping**
- **policy-attachments**
- **policy-parameters**
- **profile**
- **radius**

- **raid-activie**
- **raid-change-encryption-settings**
- **raid-delete**
- **raid-initialize**
- **raid-learn-battery**
- **raid-make-hot-spare**
- **raid-build**
- **raid-rebuid**
- **raid-reconcile-encryption-setting**s
- **raid-volume**
- **raid-volume-initialize-filesystem**
- **raid-volumne-repair-filesystem**
- **rbm**
- **refresh stylesheet**
- **refresh-tam-certs**
- **refresh-tam-keystore-pwd**
- **remove chkpoint**
- **reset**
- **reset domain**
- **reset username**
- **restart domain**
- **rmdir**
- **rollback chkpoint**
- **rule**
- **samlattrs**
- **save chkpoint**
- **save error-report**
- **save internal-state**
- **save-config overwrite**
- **scc**
- **search results**
- **send error-report**
- **send file**
- **service battery-installed**
- **service nagle**

- **set-system-var**
- **show**
- **show audit-log**
- **show audit-search**
- **shutdown**
- **simple-rate-limiter**
- **sign**
- **slm-action**
- **slm-cred**
- **slm-policy**
- **slm-rsrc**
- **slm-sched**
- **smtp-server-connection**
- **source-as1**
- **source-sa2**
- **source-as3**
- **source-ebms2**
- **source-ftp-poller**
- **source-ftp-server**
- **source-http**
- **source-https**
- **source-sftp-poller**
- **source-ssh-server**
- **ssh**
- **sshclientprofile**
- **sslproxy**
- **sskey**
- **statistics**
- **stylepolicy**
- **summary**
- **switch domain**
- **syslog**
- **switch domain**
- **system**
- **test hardware**

- **test logging**
- **test password-map**
- **test schema**
- **test tcp-connection**
- **test urlmap**
- **test urlrefresh**
- **test urlrewrite**
- **throttle**
- **timezone**
- **top**
- **traceroute**
- **tx-warn**
- **undo**
- **urlmap**
- **urlrefresh**
- **urlrewrite**
- **user**
- **user-agent**
- **user-expire-password**
- **user-password**
- **valcred**
- **validate**
- **vlan-sub-interface**
- **visible-domain**
- **watchdog**
- **web-mgmt**
- **write memory**
- **wsgw**
- **wsm-agent**
- **wsm-endpointrewrite**
- **wsm-rule**
- **wsm-stylepolicy**
- **xml parser limits**
- **xml validate**
- **xmlfirewall**

- **xml-manager**
- **xml-mgmt**
- **xpath-routing**
- **xls cache size**
- **xls checksummed cache**
- **xslconfig**
- **xslrefresh**

Appendix C: Cryptographic Stylesheets

In the evaluated configuration, the only following stylesheets can be used, unmodified, to perform cryptographic functions, such as signing or encrypting content.

Stylesheet name	Description
sign-enveloped.xsl	Creates enveloped signatures in any XML message using DSA or RSA signatures.
sign-enveloping.xsl	Creates a signature that envelops the entire message within an object node using DSA or RSA signatures.
sign-hmac-enveloped.xsl	Creates enveloped signatures in any XML message using an HMAC signature.
sign-hmac-soap-enveloped.xsl	Creates a signature that envelops the entire message within an object node using an HMAC signature.
sign-hmac-soapsec.xsl	Create a "SOAP Security" signature using an HMAC
sign-hmac-wssec.xsl	Create a WS-Security signature using an HMAC.
sign-soap-enveloped.xsl	Generate an enveloped DSA or RSA signature on the SOAP body.
sign-soap-enveloping.xsl	Generate an enveloping DSA or RSA signature on the SOAP body.
sign-soapsec-swa.xsl	Generate a DSA or RSA "SOAP Security" signature on a SOAP message with attachments.
sign-soapsec.xsl	Sign an XML message using the SOAP signature standard
sign-swa.xsl	Generate a DSA, RSA or HMAC WS-Security signature on a SOAP message with attachments.
sign-wssec.xsl	Sign an XML message using WS-Security.
verify.xsl	Verify an RSA or DSA signature
reject-weak-signatures.xsl	Prevents the use of weak signatures or hashing functions.

encrypt-soap.xsl	Encrypts a SOAP message
encrypt-wssec.xsl	Encrypts a WS-Security message
encrypt.xsl	Encrypts an XML message
decrypt.xsl	Decrypts an XML message
set-rsa-encrypt-param.xsl	Supporting stylesheet; cannot be used directly.
set-saml-decrypt-verify-param.xsl	Supporting stylesheet; cannot be used directly.
set-wssec-encrypt-param.xsl	Supporting stylesheet; cannot be used directly.
set-wssec-sign-param.xsl	Supporting stylesheet; cannot be used directly.
sign-wssec-common.xsl	Supporting stylesheet; cannot be used directly.
set-saml-decrypt-verify-param.xsl	Supporting stylesheet; cannot be used directly.
set-soap-receiver-param.xsl	Supporting stylesheet; cannot be used directly.
set-verify-param.xsl	Supporting stylesheet; cannot be used directly.

The functionality of signing can be implemented using only a few of these stylesheets. The stylesheets and the parameters for those stylesheets are listed here. A full explanation of the parameters can be found in the Parameters tables below.

Stylesheet	**Possible Parameters**
sign-soap-enveloping.xsl" SOAP Enveloping signature	c14nalg hashalg keypair keypair-cert keypair-key sigalg
sign-enveloping.xsl Raw XML Enveloping signature	c14nalg hashalg keypair keypair-cert keypair-key sigalg
sign-enveloped.xsl Raw XML Enveloped signature	XPath c14nalg hashalg id-ref-type keypair

Stylesheet	Possible Parameters
	keypair-cert keypair-key sigalg
sign-soap-enveloped.xsl Enveloped SOAPSec	actor-role-id c14nalg compatibility hashalg include-timestamp keypair-cert keypair-key security-header-layout sigalg symmetric-key-type token-reference-mechanism use-asymmetric-key use-key-derivation wss-x509-token-type output-type default
sign-soapsec-swa.xsl Enveloped SOAPSec with Attachments	c14nalg hashalg keypair keypair-cert keypair-key sigalg
sign-swa.xsl Enveloped WS-Security with Attachments using an Asymmetric key	actor-role-id c14nalg check-timestamp check-timestamp-created check-timestamp-elements compatibility enable-wssec-str-transform extra-prefix-list hashalg include-mustunderstand include-timestamp keypair keypair-cert keypair-key security-header-layout sigalg skitype swa-sign-compatibility swa-sign-transform timestamp-expiration-override

Stylesheet	Possible Parameters
	timestamp-expiration-period token-reference-mechanism use-asymmetric-key wss-x509-token-profile-1.0-keyidentifier-valuetype wssec-compatibility wssec-id-ref-type wssec-sign wssec-str-compatibility
sign-wssec.xsl Enveloped WS-Security using an Asymmetric key	actor-role-id c14nalg check-timestamp check-timestamp-created check-timestamp-elements compatibility enable-wssec-str-transform extra-prefix-list hashalg include-mustunderstand include-timestamp keypair keypair-cert keypair-key security-header-layout sigalg skitype timestamp-expiration-override timestamp-expiration-period token-reference-mechanism use-asymmetric-key wss-x509-token-profile-1.0-keyidentifier-valuetype wssec-compatibility wssec-id-ref-type wssec-sign wssec-str-compatibility
sign-swa.xsl Enveloped WS-Security with Attachments using Symmetric Key	actor-role-id algorithm c14nalg check-timestamp check-timestamp-created check-timestamp-elements compatibility"standard" dkt-label

Stylesheet	Possible Parameters
	dkt-length dkt-offset enable-wssec-str-transform hashalg hmac-sigalg include-mustunderstand include-timestamp recipient security-header-layout skitype swa-sign-compatibility swa-sign-transform symmetric-key-type timestamp-expiration-override timestamp-expiration-period token-reference-mechanism use-asymmetric-key use-key-derivation wss-x509-token-profile-1.0-keyidentifier-valuetype wssec-compatibility wssec-id-ref-type wssec-sign
sign-wssec.xsl Enveloped WS-Security using Symmetric Key	actor-role-id algorithm c14nalg check-timestamp check-timestamp-created check-timestamp-elements compatibility"standard" dkt-label dkt-length dkt-offset enable-wssec-str-transform hashalg hmac-sigalg include-mustunderstand include-timestamp recipient security-header-layout skitype symmetric-key-type timestamp-expiration-override timestamp-expiration-period token-reference-mechanism use-asymmetric-key

Stylesheet	Possible Parameters
	use-key-derivation wss-x509-token-profile-1.0-keyidentifier-valuetype wssec-compatibility wssec-id-ref-type wssec-sign

Encryption can be accomplished with the following stylesheets. Refer to the Parameters tables for an explanation of the parameters.

Stylesheet	Possible Parameters
Encrypt-wssec encrypt-soap	actor-role-id algorithm compatibility derivation-base dkt-label dkt-offset encryption-key-type include-mustunderstand include-reference-list key-transport-algorithm oaep-digest-algorithm oaep-params recipient security-header-layout swa-compatibility swa-encrypt-transform symmetric-keywrap-algo token-reference-mechanism wss-x509-token-profile-1.0-binarysecuritytoken-reference-valuetype wssc-version wssec-compatibility wssec-encrypt wssec-id-ref-type
encrypt.xsl	algorithm enctype key-transport-algorithm oaep-digest-algorithm oaep-params recipient

Stylesheet	Possible Parameters
	saml-encryption use-dynamic-enccert

Decryption can be accomplished using the decrypt.xsl stylesheet. The following table lists the possible parameters. Refer to the Parameters tables below for an explanation of the parameters.

Stylesheet	Possible Parameters
decrypt.xsl	actor-role-id asymmetric-key-encryption-algorithm decrypt-with-key-from-ed decryptkey must-be-encrypted-content must-be-encrypted-element optimize-element-decryption preserve-key-chain saml-skew-time validate-saml wssec11-enckey-cache

Verification of signatures can be accomplished with the verify.xsl stylesheet. The following table lists the possible parameters. Refer to the Parameters tables below for an explanation of the parameters.

Stylesheet	Possible Parameters
verify.xsl	actor-role-id check-signatureconfirmation check-timestamp check-timestamp-created check-timestamp-elements clientprinc enable-wssec-remote-token keytab must-be-signed-xpath remote-token-process-uri remote-token-retrieval-idcred remote-token-sslprofile saml-skew-time serverprinc signature-method-type signatureconfirmation-requested

Stylesheet	Possible Parameters
	sskey timestamp-expiration-override validate-saml wssec11-enckey-cache

Signing Parameters

The following table provides explanations for the parameters to the various signing stylesheets.

Parameter	Description	Values
use-asymmetric-key	Specifies if an asymmetric key shall be used for RSA/DSA signing or a symmetric key shall be used for HMAC signing. This setting will result in different signing algorithm and KeyInfo output. By default it is "on" meaning the RSA/DSA key is required as the default behavior for WSSec signing; otherwise a symmetric key is required for WSSec HMAC signing.	on off
c14nalg	The exclusive canonicalization algorithm for signing.	Exclusive : exc-c14n Exclusive with comments: exc-c14n-comments
hmac-sigalg	HMAC signing algorithm. The default value is hmac-sha1.	

Value	Description
sha1	http://www.w3.org/2000/09/xmldsig#sha1
sha224	http://www.w3.org/2001/04/xmldsig-more#hmac-sha224
sha256	http://www.w3.org/2001/04/xmldsig-more#hmac-sha256
sha384	http://www.w3.org/2001/04/xmldsig-more#hmac-sha384
sha512	http://www.w3.org/2001/04/xmldsig-more#hmac-sha512

Parameter	Description	Values
hashalg	The hash algorithm for the generated message digest	
	Value	Description
	sha1	http://www.w3.org/2000/09/xmldsig#sha1
	sha224	http://www.w3.org/2001/04/xmldsig-more#sha224
	sha256	http://www.w3.org/2001/04/xmldsig-more#sha256
	sha384	http://www.w3.org/2001/04/xmldsig-more#sha384
	sha512	http://www.w3.org/2001/04/xmldsig-more#sha512
symmetric-key-type	Specify what type of the symmetric key the HMAC signing will use. By default the value is "sct-available", which uses a key from a WS-SecureConversation security context. The key identified by this parameter can be used either directly as the HMAC signature key, or as the base key in a derived key scenario.	
	Value	Description
	dkt	Use an Existing DKT Token
	encryptedkey	Use a Random Key and Encrypt It for the Recipient
	static	Use Static SharedSecret Object
	eks	Use EncryptedKeySHA1 for the Recipient
	saml-symmetric-hok	Use the Symmetric SAML HoK Token from the Recipient
algorithm	The symmetric encryption algorithm to use.	
	Value	Description
	3DES-CBC	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
	AES128-CBC	http://www.w3.org/2001/04/xmlenc#aes128-cbc
	AES192-CBC	http://www.w3.org/2001/04/xmlenc#aes192-cbc
	AES256-CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc
sigalg	Signing algorithm. The default value is rsa-sha1.	
	Value	Description
	rsa-sha1	http://www.w3.org/2000/09/xmldsig#rsa-sha1

Parameter	Description	Values
	Rsa	http://www.w3.org/2000/09/xmldsig#rsa-sha1
	dsa	http://www.w3.org/2000/09/xmldsig#dsa-sha1
	dsa-sha1	http://www.w3.org/2000/09/xmldsig#dsa-sha1
	rsa-sha256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
	rsa-sha384	http://www.w3.org/2001/04/xmldsig-more#rsa-sha384
	rsa-sha512	http://www.w3.org/2001/04/xmldsig-more#rsa-sha512
keypair	The base of the names of the key and certificate to use. This value is the first part of the name used for both the key and certificate. The end part of the key's name is "KEY" and of the certificate's name is "CERT". For example, enter "foo" if the key is named "fooKEY" and the certificate is named "fooCERT". The base name may be taken from a query parameter called "dpquery:keypair" by entering the value "%url%", or from a HTTP header named "X-Use-Credentials" by entering the value "X-Use-Credentials". If the key and certificate don't follow the base name naming convention then use the separate Key and Certificate parameters instead of this Base Name parameter.	Object name
keypair-key	The key to use. Setting this overrides any value set in the Key/Certificate Base Name.	Object name
keypair-cert	The certificate to use. Setting this overrides any value set in the Key/Certificate Base Name.	Object name
wssc-version	The version of WS-Security to	1.0

Parameter	Description	Values
	use.	1.1 draft-12 draft-13
token-reference-mechanism	The method used to reference the security token in a WS-Security message.	
	Value	Description
	Direct	A BinarySecurityToken is placed in the message and a Reference element with a URI fragment pointing to the BinarySecurityToken is used to refer to it.
	KeyIdentifier	A reference is made using a KeyIdentifier element with a SubjectKeyIdentifier ValueType and content.
	ThumbPrintSHA1	A reference is made using a KeyIdentifier element with a ThumbPrintSHA1 ValueType and content. This is a WS-Security 1.1 feature. It should only be used if the "WS-Security Version" is set to at least 1.1.
	ThumbprintSHA1	A reference is made using a KeyIdentifier element with a ThumbprintSHA1 ValueType and content. This is a WS-Security 1.1 feature. It should only be used if the "WS-Security Version" is set to at least 1.1.
	X509IssuerSerial	A reference is made using an X509IssuerSerial element that identifies a certificate by its X.509 Issuer and Serial Number.
wss-x509-token-type	Determines the WS-Security X.509 Binary Security Token "TokenType" attribute for the created token	X.509 : An X.509 signature-verfication certificate. (URI #X509v3) PKCS#7 : A list of X.509 certificates and (optionally) CRLs packaged in a PKCS#7 wrapper. (URI #PKCS7) PKIPath : An ordered list of X.509 certificates that are packaged in a PKIPath. (URI #X509PKIPathv1)
wss-x509-token-profile-1.0-binarysecuritytoken-reference-valuetype	Controls the value of BinarySecurityToken/@ValueType and of SecurityTokenReference/Reference/@ValueType when referring to a BinarySecurityToken that is an X.509 token. Because of the differences between the final WS-Security 1.0 standards and	#X509 : Generates http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509 #X509v3 : Generates http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

Parameter	Description	Values
	the multiple draft versions of its errata document, different values of the ValueType attribute are used by different WS-Security implementations. Compatibility with certain versions of .NET Web Services Enhancements (WSE) might require setting this option to #X509v3. Compatibility with certain versions of WebSphere might require setting this option to #X509. This setting is relevant only if the X.509 token type parameter is X.509.	
include-timestamp	Setting to 'on', the default, causes the output message to include a Timestamp block.	On off
timestamp-expiration-period	The expiration period in seconds for the Timestamp (and therefore of the security semantics in this signature). A value of zero (0) means no expiration. The default is 300 seconds (5 minutes). The maximum is 31536000 seconds (365 days).	Integer 0 - 31536000
check-timestamp-elements	Setting to 'on', the default, causes an existing Timestamp block to be validated for the number of Timestamp blocks, 'Created' and 'Expires' elements. Setting it to 'on' also enables to control the checking of Created and Expiration times. See 'Check Timestamp Created' and 'Check Timestamp Expiration' toggles. This setting applies to the time range specified by NotBefore and NotOnOrAfter of a saml:Conditions. Setting to 'off' prevents checking	On off

Parameter	Description	Values
	timestamp blocks for any errors.	
check-timestamp-created	Setting to 'on', causes an existing Timestamp block to be checked for created time. It should always be lesser than the current time. If not, the transaction is terminated. This toggle is activated only when the toggle 'Check Timestamp' is set to 'on'. Setting to 'off' prevents checking Timestamp Created.	On off
check-timestamp	Setting to 'on', the default, causes an existing Timestamp block to be checked for expiration when an expiration time is specified, and the transaction terminated if the Timestamp is expired. This toggle is activated only when the toggle 'Check Timestamp' is set to 'on'. Setting to 'off' prevents checking Timestamp expiration.	On off
timestamp-expiration-override	The override expiration period in seconds for the Timestamp checking. A value of zero (0) means no override. The default is 0. The maximum is 630720000 seconds (20 years).	Integer 0 - 630720000
sign-binarysecuritytoken	If the Token Reference Mechanism is "Direct" then by default the inserted BinarySecurityToken is not signed. Setting this switch to 'on' causes the BinarySecurityToken to be signed. In other words, the digital signature will cover the BinarySecurityToken along with the other signed portions of the message. Compatibility with certain versions of BEA	On off

Parameter	Description	Values
	WebLogic may require setting this parameter to 'on'.	
signature-idcred	The identity credential used to generate the signature. This is currently only applicable when the BinarySecurityToken ValueType is either "#PKCS7" or "#X509PKIPathv1". The valcred key is used to generate the signature itself, and the associated certificates are placed in the WS-Security BinarySecurityToken in the chosen encoding format.	Name of object
include-signatureconfirmation	SignatureConfirmation only applies to WS-Security 1.1. Setting this switch to 'on' causes SignatureConfirmation to be generated if the request contains "ds:SignatureValue".	On off
expect-signatureconfirmation	If we expect the returned response message contains WS-Security 1.1 SignatureConfirmation, set this switch to 'on' to save the generated signature value, so that a Verify action can process the response to verify the WS-Security 1.1 SignatureConfirmation.	On off
security-header-layout	The layout rule to apply to the security header.	strict : Items are added to the security header following the numbered layout rules described in the WS-Security Policy specification. lax : Items are added to the security header in any order that conforms to WSS:SOAP Message Security. laxtimestampfirst : As Lax except that the first item in the security header MUST be wsse:Timestamp. laxtimestamplast : As Lax except that the last item in the security header MUST be wsse:Timestamp.
enable-wssec-str-transform	If the target to be signed has a wsse:SecurityTokenReference, especially for the field level	On off

Parameter	Description	Values
	wssec signing, the STR Dereference Transform (STRDT) can be used to sign the security token that the STR pointing at rather the STR element itself.	
extra-prefix-list	By default, no value is needed and all the visibly utilized namespaces are protected. Otherwise, specify the namespace prefixes that are additionally signed with the target nodeset. It is strongly recommended to put empty string for this setting and have better compatibility for exclusive canonicalization. This setting is enabled only when the STR-Transform is used per some special requirement for WAS. The signature with exclusive canonicalization transformation algorithms always covers the namespaces that are being visibly utilized by the target element or attribute. When a namespace is inherited from parent or ancestor elements and not being visibly utilized by the target nodes, this setting can be used to include that namespace and protect that namespace and its URI. The #default prefix string is defined by the specification to include the namespace has no prefix.	Namespace name
wssec-str-compatibility	Select what type of Reference is used by the WS Security Token Reference for the STR-Transform.	

Value	Description
standard	Use the default STR format that is standard or determined mostly compatible to the input message and configuration settings.

Parameter	Description	Values
	direct	The STR will try to use the WS-Sec Direct references, which may not be compatible to a standard or specification.
	keyid	The STR will try to use the WS-Sec Key Identifier references, which may not be compatible to a standard or specification. In case of generating STR-Transform signed local SAML 2.0 assertion for WAS 7, it is required to use this option.
sign-keyinfo	Setting to 'on' will also sign the STR element inside of the dsig:KeyInfo generated by this action. As required by WS-SecurityPolicy, when ws-sec sign action generates a BST being used by the KeyInfo, the STR for this case must be signed with STR-Transform,	On off
dkt-length	This setting indicates the size of the derived key.	Integer
dkt-offset	This setting indicates where of the derived key starts in the byte stream of the lengthy generated key sequence. The default value is zero. This setting is exclusive with the "Generation" setting described as following. Set this setting as an empty string to enable the "Generation" setting.	Integer
dkt-generation	If a fixed sized key is generated, then this optional setting can be used to specify which generation of the key to use. The value of this setting is an unsigned long value indicating the index number of the fixed key in the lengthy key sequence. It starts with zero. If this setting is set, it precedes the above "Offset" setting; that is: offset = (generation) * length	Integer

Parameter	Description	Values
dkt-label	Specify the label string for the wsc:DerivedKeyToken, if not specified, the default "WS-SecureConversationWS-SecureConversation" (represented as UTF-8 octets) is used.	Label
recipient	It is only visible when deriving a key from an "encryptedkey". When the symmetric key type is "encryptedkey", a random key is used as symmetric key which can also be used as the shared secret if key derivation is enabled. That random key will then be encrypted and returned as a xenc:EncryptedKey to the recipient. The one who verifies the signature will essentially decrypt the key before using it as the symmetric key. Specify a CryptoCertificate object, with the public certificate of the intended recipient who will verify the signed message.	Name of object
use-key-derivation	Specifies if the HMAC signing key is a derived key or not. If it is 'on', the retrieved key from the symmetric key source will be used as the key derivation base and the derived key is is the actual HMAC signing key. In this case a wsc:DerivedKeyToken will always accopany with the signature KeyInfo. By default it is "off" meaning the key from the symmetric key source will be directly used as the HMAC symmetric key. Please note: the static SSKey can not be used directly as a key derivation base as the WS-SC spec requires to put a	On off

Parameter	Description	Values
	wsse:SecurityTokenReference for DKT and the SSKey dsig:KeyName can not be referred by this reference mechanism.	
key	The name of the shared secret key to use. The name may be taken from a query parameter called "dpquery:key" by entering here the value "%url %", or from a HTTP header named "X-Use-Credentials" by entering here the value "X-Use-Credentials". This parameter is overriden by the Shared Secret Key parameter.	Name of object
base-dkt-name	When the symmetric signing key is obtained from a named DerivedKeyToken (DKT), this parameter specifies the token's name. A named DKT typically has a "wsc:Properties/wsc:Name" element.	Name
skitype	The form of the Subject Key Identifier to use. This parameter is only relevant when the WS-Security Version is 1.0/1.1 and the Token Reference Mechanism is "KeyIdentifier".	ms-wse-1 pkix
include-inline-cert	Setting to 'on' causes the signer's certificate to be included in the Signature element inside a second KeyInfo block. This may aid compatibility with certain applications.	On off
include-second-id	Setting to 'on' causes the output message to include a plain "id" attribute on the SOAP Body element in addition to the	On off

Parameter	Description	Values
	normal "wsu:Id" attribute. This may aid compatibility with certain applications.	
wssec-sign	Setting what WS-Security data will be signed: Both SOAP message and attachments, or SwA attachments only.	Message-attachments : message and attachments attachments : just the attachments
swa-sign-compatibility	Defaults to 1.1	1.1 1.0
swa-sign-transform	Defaults to MIMEConentOnly	MIMEContentOnly MIMEContentAndHeader
actor-role-id	Specify the identifier of the SOAP 1.1 actor or SOAP 1.2 role that this action work as in processing a SOAP header. Some well-known values are: http://schemas.xmlsoap.org/soap/actor/next Every one, including the intermediary and ultimate receiver, receives the message should be able to processing the SOAP header. http://www.w3.org/2003/05/soap-envelope/role/none No one should process the SOAP Header. http://www.w3.org/2003/05/soap-envelope/role/next Every one, including the intermediary and ultimate receiver, receives the message should be able to processing the SOAP header. http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver The message ultimate receiver can process the SOAP header. This is the default value if such setting is not configured. <blank or empty string> The empty string "" (without quotes) indicates that no "actor/role" identifier is configured. if there is no actor/role setting configured, the ultimateReceiver is assumed when processing the message. USE_MESSAGE_BASE_URI The value "USE_MESSAGE_BASE_URI" without quotes indicates that the actor/role identifier will be the base url of the message, if the SOAP message is transported using HTTP, the base URI is the Request-URI of the http request. any other customized string You can input any string to identify the SOAP header's actor or role.	
XPath	Enter in the blank field in front of the Add button with the XPath expression identifying the elements on which to sign. Click Add to add the expression to those included in the map. If no XPath Expression is defined, the whole xml doc will be signed. Click the XPath Too	Xpath expression
id-ref-type		Xml:id\ root

Encryption Parameters

The following table provides explanations of the possible parameters to the various encryption-related stylesheets.

Parameter	Description	Values
encryption-key-type	Specify what the bulk encyption key is and how it is protected.	Asymmetric an ephemeral key transported by X509 key/cert pair with an asymmetric algorithm symmetric ephemeral key being encrypted by a symmetric key from a security token keywrap session key protected by a security token
	The symmetric encryption algorithm to use.	

Value	Description
3DES-CBC	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
AES128-CBC	http://www.w3.org/2001/04/xmlenc#aes128-cbc
AES192-CBC	http://www.w3.org/2001/04/xmlenc#aes192-cbc
AES256-CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc

Parameter	Description	Values
key-transport-algorithm	The key transport algorithm to use for encrypting the symmetric key.	Rsa-pkcs1 rsa-oeap
oeap-params	A base64-encoded string containing the OAEP Parameters	
oeap-digest-algorithm	The message digest algorithm to use during OAEP padding	

Value	Description
sha1	http://www.w3.org/2000/09/xmldsig#sha1
sha224	http://www.w3.org/2001/04/xmldsig-more#hmac-sha224
sha256	http://www.w3.org/2001/04/xmldsig-more#hmac-sha256
sha384	http://www.w3.org/2001/04/xmldsig-more#hmac-sha384
sha512	http://www.w3.org/2001/04/xmldsig-more#hmac-sha512

Parameter	Description	Values
use-key-derivation	Specifies if a derived key will be used as the symmetric key for data encryption, or symmetric key to wrap the ephemeral key. If it is 'on', the retrieved key from the symmetric key source will be used as the key derivation base and the derived key is is the actual key to encrypt the data or wrap up the symmetric key. By default it is "off" meaning the key from the symmetric key source will be directly used for encryption. Please note: the static SSKey can not be used directly as a key derivation base as the WS-SC spec requires to put a wsse:SecurityTokenReference for DKT and the SSKey	On off
symmetric-key-type	Specify what type of the symmetric key the HMAC signing will use. By default the value is "sct-available", which uses a key from a WS-SecureConversation security context. The key identified by this parameter can be used either directly as the HMAC signature key, or as the base key in a derived key scenario.	

Value	Description
sct-available	Use a Key in Security Context
dkt	Use an Existing DKT Token
encryptedkey	Use a Random Key and Encrypt It for the Recipient
static	Use Static SharedSecret Object
eks	Use EncryptedKeySHA1 for the Recipient
saml-symmetric-hok	Use the Symmetric SAML HoK Token from the Recipient

Parameter	Description	Values
derivation-base	Select what type of key derivation to use or choose what the base token to derive a key. By default the value is "sct-available", which derives a key from an existing wsc:SecurityContextToken or fall onto "asymmetric" Encryption Key Type if no SCT token is available. If a key derivation is used by this action, a DKT is issued with the encrypted message.	
	Value	**Description**
	sct-available	Use a Key in Security Context
	dkt	Use an Existing DKT Token
	encryptedkey	Use a Random Key and Encrypt It for the Recipient
	static	Use Static SharedSecret Object
	eks	Use EncryptedKeySHA1 for the Recipient
	saml-symmetric-hok	Use the Symmetric SAML HoK Token from the Recipient
enctype	Encrypt the content of the whole message or just an element of the message	content element
saml-encryption	Setting to 'on' will generate the EncryptedAssertion and/or EncryptedAttribute elements if it is a SAML 2.0 Assertion or Attribute. This feature is effective only for the "element" encryption type. For SAML1.x message, the SAML schema doesn't define EncryptedAssertion or EncryptedAttribute types, so the standard XML Encryption will be applied by generating EncryptedData directly.	On off
use-dynamic-enccert	Enable this property to encrypt the message with the verified signing certificate. The verified signing certificate is from the preceding verify action. If the message is not signed, encrypts	On off

Parameter	Description	Values
	with the public key for the intended recipient.	
one-key-encryption	Setting to 'on' causes all the encryption in this step to use the same Ephemeral Key. There will be only one ephemeral key encryption. Its corresponding EncryptedKey will add a DataReference URI for each EncryptedData. Enabling this setting will get better performance.	On off
symmetric-keywrap-algo	When the bulk encryption key is itself encrypted by a symmetric key, this parameter determines the key wrap algorithm used.	Kw-tripledes kw-aes128 kw-aes192 kw-aes256
validate-saml	Validate the SAML assertion used by the crypto operation	On off
saml-skew-time	Skew time is the difference, in seconds, between the device clock time and other system times. When the skew time is set, the SAML assertion expiration takes the time difference into account when the appliance consumes SAML tokens. NotBefore is validated with CurrentTime minus SkewTime. NotOnOrAfter is validated with CurrentTime	integer
token-reference-mechanism	The method used to reference the security token in a WS-Security message.	

Value	Description
Direct	A BinarySecurityToken is placed in the message and a Reference element with a URI fragment pointing to the BinarySecurityToken is used to refer to it.
KeyIdentifier	A reference is made using a KeyIdentifier element with a SubjectKeyIdentifier ValueType and content.
ThumbPrintSHA1	A reference is made using a KeyIdentifier element with a ThumbPrintSHA1 ValueType and content.

Parameter	Description	Values
		This is a WS-Security 1.1 feature. It should only be used if the "WS-Security Version" is set to at least 1.1.
	ThumbprintSHA1	A reference is made using a KeyIdentifier element with a ThumbprintSHA1 ValueType and content. This is a WS-Security 1.1 feature. It should only be used if the "WS-Security Version" is set to at least 1.1.
	X509IssuerSerial	A reference is made using an X509IssuerSerial element that identifies a certificate by its X.509 Issuer and Serial Number.
key	The name of the shared secret key to use. The name may be taken from a query parameter called "dpquery:key" by entering here the value "%url %", or from a HTTP header named "X-Use-Credentials" by entering here the value "X-Use-Credentials". This parameter is overriden by the Shared Secret Key parameter.	Object name
recipient	It is only visible when deriving a key from an "encryptedkey". When the symmetric key type is "encryptedkey", a random key is used as symmetric key which can also be used as the shared secret if key derivation is enabled. That random key will then be encrypted and returned as a xenc:EncryptedKey to the recipient. The one who verifies the signature will essentially decrypt the key before using it as the symmetric key. Specify a CryptoCertificate object, with the public certificate of the intended recipient who will	Name of object

Parameter	Description	Values
	verify the signed message.	
dkt-offset	This setting indicates where of the derived key starts in the byte stream of the lengthy generated key sequence. The default value is zero. This setting is exclusive with the "Generation" setting described as following. Set this setting as an empty string to enable the "Generation" setting.	Integer
dkt-generation	If a fixed sized key is generated, then this optional setting can be used to specify which generation of the key to use. The value of this setting is an unsigned long value indicating the index number of the fixed key in the lengthy key sequence. It starts with zero. If this setting is set, it precedes the above "Offset" setting; that is: offset = (generation) * length	Integer
dkt-label	Specify the label string for the wsc:DerivedKeyToken, if not specified, the default "WS-SecureConversationWS-SecureConversation" (represented as UTF-8 octets) is used.	Label
base-dkt-name	When the symmetric signing key is obtained from a named DerivedKeyToken (DKT), this parameter specifies the token's name. A named DKT typically has a "wsc:Properties/wsc:Name" element.	Name
skitype	The form of the Subject Key Identifier to use. This parameter is only relevant	ms-wse-1 pkix

Parameter	Description	Values
	when the WS-Security Version is 1.0/1.1 and the Token Reference Mechanism is "KeyIdentifier".	
include-reference-list	Specify if the Security header will include the xenc:ReferenceList element or not.	On off
wssec11-enckey-cache	This is the Cache Lifetime for the generated key. Setting the value to 0 means the generated key will not be cached.	Integer
include-sct-token		On off
swa-sign-compatibility	Defaults to 1.1	1.1 1.0
swa-sign-transform	Defaults to MIMEConentOnly	MIMEContentOnly MIMEContentAndHeader

Decryption Parameters

The following table provides explanations for the parameters to the decrypt.xsl stylesheet.

Parameter	Description	Values
asymmetric-key-encryption-algorithm	If a value is not explicitly selected, all asymmetric key XML encryption algorithms will be permitted. If a particular encryption algorithm is selected, all <xenc:EncryptedKey> elements must be encrypted using this algorithm (assuming asymmetric encryption), or the request will be rejected.	Rsa-pkcs1 rsa-oeap
bulk-encryption-algorithm	If a value is not explicitly selected, all bulk XML encryption algorithms will be permitted. If a particular encryption algorithm is selected, all <xenc:EncryptedData> elements must use this algorithm, or the request will be rejected.	

Descriptoion	Value
3DES-CBC	http://www.w3.org/2001/04/xmlenc#tripledes-cbc
AES128-CBC	http://www.w3.org/2001/04/xmlenc#aes128-cbc
AES192-CBC	http://www.w3.org/2001/04/xmlenc#aes192-cbc
AES256-CBC	http://www.w3.org/2001/04/xmlenc#aes256-cbc

decrypt-with-key-from-ed	In scenarios where the key is inside an EncryptedData element (such as 'encrypted SAML Assertion'), the decrypt action cannot locate the key to decrypt the corresponding EncryptedData elements. Select 'on', to enable decrypt action to attempt decryption with the key that is inside the EncryptedData element. The default is 'off'.	On off
decryptkey	The CryptoKey object which will be used as the private key to decrypt the encrypted data. This parameter is optional. If the value is not empty, it assumes the explicit CryptoKey object will be used	Name of object

Parameter	Description	Values
	to decrypt all the encrypted data in the input message. Otherwise, either the decryption Algorithm doesn't need an explicit CryptoKey object, or the Algorithm can get the private key indirectly by reverse look-up of the key based on the certificate. For both cases, you may define Crypto Identification Credential objects accordingly.	
must-be-encrypted-content	If any XPath expressions are configured for this paramter, the nodesets which result from the expression must be encrypted using #Content-style XML Encryption. So, the message must contain the nodeset resulting from the XPath expression, and the elements in the nodeset must be encrypted	Xpath expression
must-be-encrypted-element	If any XPath expressions are configured for this paramter, the nodesets which result from the expression must be encrypted using #Element-style XML Encryption. So, after decryption, either the exact XPath nodeset or an ancestor of the nodeset must have been encrypted in the original message.	Xpsth expression
optimize-element-decryption	According to the encryption specifications the result of decrypting data which is "element" encrypted (versus "content" encrypted) should be valid XML. This means it should contain all namespace prefix bindings needed to parse the resulting XML data. If you	On off

Parameter	Description	Values
	know the source of the encrypted data follows this practice then setting this parameter to 'on' may improve decryption performance since extra canonicalization during decryption is not required. However, for compatibility it may be necessary to set this parameter 'off' since some XML encryption devices may not follow the rules for preparing data before encrypting it. The default is 'off'.	
preserve-key-chain	Select 'on' to output the chain of elements being used by the decrypted Encrypted Data, such as xenc:EncryptedKey, wsc:DerivedKeyToken. Otherwise all the xenc:EncryptedKey elements will be removed after decryption, even when some of the Encrypted Data may not be decrypted successfully. The default is 'off'.	On off
saml-skew-time	Skew time is the difference, in seconds, between the device clock time and other system times. When the skew time is set, the SAML assertion expiration takes the time difference into account when the appliance consumes SAML tokens. NotBefore is validated with CurrentTime minus SkewTime. NotOnOrAfter is validated with CurrentTime plus SkewTime.	Integer
symmetric-key-encryption-algorithm	If a value is not explicitly selected, all symmetric key XML encryption algorithms will be permitted. If a particular encryption algorithm is selected, all <xenc:EncryptedKey> elements must be encrypted using this algorithm (assuming symmetric encryption), or the request will be rejected.	

Parameter	Description	Values
	Possible values: http://www.w3.org/2001/04/xmlenckw-tripledes http://www.w3.org/2001/04/xmlenc#kw-aes128 http://www.w3.org/2001/04/xmlenc#kw-aes192 http://www.w3.org/2001/04/xmlenc#kw-aes256	
validate-saml	Validate the SAML assertion used by the crypto operation	On off
wssec11-enckey-cache	This sets the Cache Lifetime for the extracted key material being used during the Decrypt operation. The key material might be an EncryptedKey token, an EncryptedKeySHA1 key identifier, or a SAML token, which, if the response must be encrypted/signed, must be put in the cache by using the same key and/or key referencing, such as with EncryptedKeySHA1 or SAML. If the value is set to 0, the extracted key material will not be cached.	Integer

Verification Parameters

The following table provides explanations for the parameters to the verify.xsl stylesheet.

Parameter	Description	Values
actor-role-id	Specify the identifier for the SOAP1.1 actor or SOAP1.2 role in processing a WS-Sec Security Header. This is only effective when a SOAP message is being used for WS-Security 1.0/1.1.	
check-signatureconfirmation	This will check for SignatureConfirmation according to the requirement specified in WS-Security 1.1. By setting it to 'on', if no SignatureConfirmation is is given in the message, the message will fail the verify step.	On off
check-timestamp	Setting to 'on', the default, causes an existing Timestamp block to be checked for expiration when an expiration time is specified, and the transaction terminated if the Timestamp is expired. This toggle is activated only when the toggle 'Check Timestamp' is set to 'on'. Setting to 'off' prevents checking Timestamp expiration.	On off
chec-timestamp-created	Setting to 'on', causes an existing Timestamp block to be checked for created time. It should always be lesser than the current time. If not, the transaction is terminated. This toggle is activated only when the toggle 'Check Timestamp' is set to 'on'. Setting to 'off' prevents checking Timestamp Created.	On off
check-timestamp-elements	Setting to 'on', the default, causes an existing Timestamp block to be checked for expiration when an expiration time is specified, and the transaction terminated if the Timestamp is expired. This	On off

Parameter	Description	Values
	toggle is activated only when the toggle 'Check Timestamp' is set to 'on'. Setting to 'off' prevents checking Timestamp expiration.	
timestamp-expiration-override	The override expiration period in seconds for the Timestamp checking. A value of zero (0) means no override. The default is 0. The maximum is 630720000 seconds (20 years).	integer
enable-wssec-remote-token	The WS-Security 1.1 profiles, such as SAML Token Profile, specifies a mechanism to refer to the special remote tokens, which can be retrieved if this setting is 'on'	On off
must-be-signed-xpath	If any XPath expressions are configured for this parameter, the nodesets which result from the expression, if exist, must be signed. So, a signature reference must either sign the result of the XPath expression, or an ancestor of the expression nodeset. This configuration parameter may be used to express the contents of a SignedElements WS-SecurityPolicy assertion.	Xpath expression
remote-token-process-uri	The WS-Security 1.1 profiles, such as SAML Token Profile, specifies a mechanism to refer to the special remote tokens, which must be retrieved in order to verify its signature. The remote WS-Sec token could be signed, encrypted or encoded. A firewall or proxy service with different actions can be used to process the remote token, either decrypting pieces of a remote SAML	URI

Parameter	Description	Values
	assertion, doing a xslt transform, or using AAA to assert the token. This setting is the URL for that service, which accepts the security token as the request of the SOAP call, and provides the final security token as the response if successful.	
remote-token-retrieval-idcred	The WS-Security 1.1 profiles, such as SAML Token Profile, specifies a mechanism to refer to the special remote tokens, which must be retrieved in order to verify its signature. If an identity credential is configured for this action, it will be used to sign the SAML assertion retrieval message	Object name
remote-token-sslprofile	The WS-Security 1.1 profiles, such as SAML Token Profile, specifies a mechanism to refer to the special remote tokens, which must be retrieved in order to verify its signature. If the remote side requires secure socket connection, this setting can be specified with the corresponding SSLProxyProfile object.	Object name
restrict-algorithm	Setting this configuration parameter to "on" will require all signatures to be signed using the RSA SHA1 XML signature algorithm, whose URI is http://www.w3.org/2000/09/xmldsig#rsa-sha1.	On off
saml-skew-time	Skew time is the difference, in seconds, between the device clock time and other system times. When the skew time is set, the SAML assertion	Integer

Parameter	Description	Values
	expiration takes the time difference into account when the appliance consumes SAML tokens. NotBefore is validated with CurrentTime minus SkewTime. NotOnOrAfter is validated with CurrentTime plus SkewTime.	
signature-method-type	This identifies what type of signatures will be verified or what type of signatures will be ignored instead. If the message contains signatures signed by RSA/DSA (asymmetric) and HMAC (symmetric) algorithms, this setting can be configured to verify only just one type of signing algorithms and ignore the other type algorithms, or verify all signatures. By default, this action verifies only RSA or DSA signatures using asymmetric algorithm.	All Signatures : Verify all signatures. HMAC Signatures : Verify only the signatures with symmetric signing methods. Verification fails if an RSA/DSA signature is found. RSA/DSA Signatures : Verify only the RSA/DSA signatures with asymmetric signing methods. Verification fails if an HMAC signature is found.
signer	This identifies the certificate of the signer, which is used to verify the asymmetric signature. If this is left blank, the certificate information is taken from the WS-Sec token or the signature information, which is the standard case. Alternately, the name of the certificate may be taken from a query parameter called "dpquery:signer" by entering here the value "%url%". Query parameters are defined in the service. The signer certificate may also be explicitly specified using the form	Name of object

Parameter	Description	Values
	"name:CryptoCertificateObject". For example "name:alice" specifies to user the Crypto Certificate object called "alice" to verify the signature.	
valcred	The Validation Credential to use for validating the signer's certificate.	Name of object
validate-saml	Validate the SAML assertion used by the crypto operation	On off
wssec11-enckey-cache	This sets the Cache Lifetime for the extracted key material being used during the Verify operation. The key material might be an EncryptedKey token, an EncryptedKeySHA1 key identifier, or a SAML token, which, if the response must be encrypted/signed, must be put in the cache by using the same key and/or key referencing, such as with EncryptedKeySHA1 or SAML. If the value is set to 0, the extracted key material will not be cached.	Integer

Notices

Trademarks