April 2013

IBM WebSphere Application Server Version 8 for Linux on IBM System z – SSL Setup and Performance Study

Table of Contents

About this publication

This paper provides an overview of the SSL performance for the encrypted communication with an IBM WebSphere® Application Server (WAS) exploiting IBM® System z® cryptographic hardware features.

Two possible WAS setup scenarios are discussed:

- The first scenario describes a WAS-only setup
- The second scenario involves an IBM HTTP server (IHS) as HTTP transport service for WAS

The focus of this white paper is on the support and setup of IBM System z cryptographic hardware features for both scenarios. The SSL performance results for an online stock trading benchmark application are also discussed.

Authors

- Thomas Weber
- Dr. Juergen Doelle

Acknowledgements

Thank you to the following people for their support during this project:

- Michael Tebolt (IBM Poughkeepsie)
- Ingo Tuchscherer (IBM Boeblingen)

The tests for this project were performed at the IBM System z End to End Performance Team in Boeblingen, Germany.

Remarks

IBM online information center Web links that point to specific application documentation chapters are very long. Therefore the following notation is used in this white paper. The Web link points to the top level site of the application documentation and the navigator below the Web link indicates the path to a specific topic.

For example:
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

Websphere Application Server (Distributed operating systems), Version 8.0 > Reference > Custom properties

Introduction

This paper describes the experiences using, and measurement results for, an end-to-end performance software stack with IBM WebSphere Application Server (WAS) Version 8 (64-bit) for Linux® on System z and IBM DB2® Enterprise Server Edition, Version 9.7 (64-bit).

The Java™ 2 Platform Enterprise Edition (J2EE) benchmark application running on the application server emulates an online stock trading system.

A second scenario involves the IBM HTTP server (IHS) for WAS Version 8, which runs in addition to the application server acting as HTTP transport for WAS.

The scenarios focus on SSL secured communication between the WAS-hosted Web application and client hosts issuing stock trading queries against the Web server application. Hence the SSL secured network connection between the client and the application server is the main focus for the System Under Test (SUT).

There are two scenarios considered for this study:

- Securing WAS internal HTTP transport using SSL
- Securing IHS HTTP server transport using SSL

IBM Websphere Application Server Version 8

IBM WebSphere Application Server (WAS) is IBM's leading open standards-based application foundation.

It is based on open standards such as Java Platform, Enterprise Edition (Java EE), XML, and Web services and it provides a service-oriented architecture (SOA) platform to host, for example, applications that conform to Java EE.

The security model for WAS uses services offered by the underlying operating system and the Java EE security model.

IBM System z cryptographic hardware features

The IBM zEnterprise® 196 (z196) used in this scenario was equipped with two cryptographic hardware features.

- CP Assist for Cryptographic Function (CPACF) is a feature on the processor unit (cryptographic feature code 3863). It accelerates symmetric cryptographic functions (DES, 3DES, AES). The Secure Sockets Layer (SSL) protocol uses symmetric cipher encryption/decryption operations during the network data transmission after the SSL connection has been established. CPACF also supports SHA hash functions (SHA-1, SHA-256). The available WAS and IHS cipher suites involve symmetric ciphers as well as hash functions.
- The Crypto Express (CEX) feature complements the System z cryptographic features. Crypto Express3 (CEX3) provides improved performance for asymmetric operations. It supports clear key and secure key modes. The feature is capable of offloading and accelerating RSA ciphers. RSA ciphers are basic parts of SSL cipher suites and are used in the SSL handshake process when initiating a SSL connection. RSA is a common algorithm used in public key cryptography. The RSA cipher is the most CPU-intensive operation whenever a network connection is secured using SSL.
- An adapter on a Crypto Express feature can be either defined as a cryptographic coprocessor (CEX3C) or as a cryptographic accelerator (CEX3A). Both types are considered in the cryptographic setup discussed in this paper.

Note: **Terminology:** An IBM System z cryptographic feature has a feature code because it is an orderable entity. For Crypto Express (CEX) the functional entity is called an adapter. A Crypto Express3 (CEX3) feature contains two CEX adapters. CPACF is also considered as a feature, because it is an orderable (non-priced) entity and has a feature code.

Objectives

This study examines the exploitation and advantages of using the IBM System z cryptographic hardware features for accelerated SSL connections (using clear key).

The cryptographic features available for this test scenario are CP Assist for Cryptographic Function (CPACF) and Crypto Express (CEX).

The setup part of this paper outlines the required customization steps required to enable SSL support for IBM WebSphere Application Server (WAS) in a first scenario and for IBM HTTP server (IHS) in a second scenario. It describes how to enable the cryptographic hardware support that is available on the IBM System z platform.

The following configurations are considered in more detail:

- WAS SSL setup for securing network communications
- IHS SSL setup for securing network communications
- Linux on System z cryptographic configurations required

The results part discusses the SSL performance for both setups using the different levels of System z cryptographic hardware features.

The SSL performance results include:

- Results for SSL cryptographic operations in software only
- Results for SSL cryptographic operations supported by CPACF
- Results for SSL cryptographic operations supported by CPACF and CEX3
  - CEX3 configured as SSL Accelerator (CEX3A)
  - CEX3 configured as co-processor (CEX3C)
- Results for different RSA key lengths\

Summary

The correct setup, which has cryptographic hardware support enabled can greatly improve throughput rates and achieve a good SSL performance with IBM WebSphere Application Server (WAS) or IBM HTTP server (IHS) as web server frontend.

When System z cryptographic features are available, it is recommended to use them in order to achieve the best throughput and response times for a SSL secured communication.

One important aspect is the selection of an appropriate SSL cipher suite that is supported by IBM System z cryptographic features, for example a cipher suite consisting of RSA, AES and SHA algorithms. The additional effort required by the SSL protocol causes extra workload on the WAS server driving the SSL encryption, no matter whether WAS or the IHS is managing the SSL connection. The faster and less CPU intensive this additional workload can be processed on the general purpose CPUs, the faster is the response time and the more user queries can be processed at the same time.

The stronger a selected cipher suite for SSL is, the more CPU power is required on the WAS server. Especially when using RSA key sizes of 2048-bit or 4096-bit, IBM System z cryptographic features contribute to provide good performance and response times on the WAS server. A RSA key size of 1024-bit is no longer recommended for security reasons.

Adding the IHS to the system under test (SUT) (scenario 2) results in better transaction throughput numbers for all the considered cryptographic setups.

IBM System z hardware and software used

This section provides an overview of the software and hardware configurations used to set up the system under test (SUT).

The SUT forms a three tier architecture consisting of an application server, a database server and a client machine.

It is implemented on two LPARs on an z196 (Model 2817-M66) as follows:

- One for the application server and the IBM HTTP server (IHS) web server
- One for the database server

The SUT used a IBM System Storage® DS8800 subsystem connected with 8 FICON® Express8 features, shared with both LPARs.

IBM Websphere Application Server

The tables in this topic list the hardware and software used for the IBM WebSphere Application Server (WAS) setup in a IBM System z LPAR.

Table 1. Application Server software

Software	Service level
▪ Novell SUSE Linux Enterprise Server 11 for System z (64-bit) ▪ IBM Websphere Application Server Version 8.0 ▪ IBM HTTP Server for WAS Version 8.0	▪ Service Pack 2 (SP2) ▪ Fix Pack 2 for WAS (8.0.0.2) ▪ Fix Pack 2 for WAS supplements (8.0.2)

Table 2. Application Server hardware

Server hardware	Setup
IBM z196 LPAR (Model 2817-M66) IBM System Storage DS8800	▪ 4 IFLs (CPUs) ▪ 8 GB central storage ▪ 2 x Crypto Express3 (CEX3) features – 1 adapter configured as CEX3A – 1 adapter configured as CEX3C ▪ 10 GbE OSA-Express3 for client connectivity ▪ 1 GbE OSA-Express2 for administration ▪ HiperSockets™ (8k MTU size) for database server connection ▪ FICON Express8 feature ▪ 1x DASD Model 27

Database server

The tables in this topic list the hardware and software used for the DB2 database server setup in a IBM System z LPAR.

Table 3. Database server software

Software	Service level
▪ Novell SuSE Linux Enterprise Server 11 for System z (64-bit) ▪ IBM DB2 Enterprise Server Edition Version 9.7	▪ Service Pack 2 (SP2) ▪ Fix Pack 4 (9.7.0.4)

Table 4. Database server hardware

Server hardware	Setup
IBM z196 LPAR (Model 2817-M66) IBM System Storage DS8800	▪ 4 IFLs (CPUs) ▪ 8 GB central storage ▪ 1 GbE OSA-Express2 for administration ▪ HiperSockets (8k MTU size) for application server connection ▪ FICON Express8 feature ▪ 1 x DASD Model 27 ▪ 4 HyperPAV aliases

Client used as workload driver

The tables in this topic list the client hardware and software used to set up the workload driver for the System Under Test (SUT).

Table 5. Client software

Software	Service level
▪ Novell SUSE Linux Enterprise Server 11 ▪ Websphere Studio Workload Simulator (workload driver)	▪ Service Pack 1 (SP1)

Table 6. Client hardware

Client hardware	Setup
IBM eServer™ xSeries® 445 (x445)	▪ 8 CPUs (2.7 GHz) ▪ 8 GB memory ▪ 10 GbE network card for application server connectivity

System under test (SUT) overview

This section describes the two IBM WebSphere Application Server (WAS) scenarios and how their software and hardware components interact.

The first scenario is a WAS only setup. WAS uses its own internal HTTP transport service in this case. The second scenario uses IBM HTTP server (IHS) as HTTP transport instead. Both scenarios are configured to support a SSL connection between the client and the application server. For the second scenario IHS is configured for SSL support.

WAS provides the application logic layer in a three-tier architecture model, enabling client components to interact with data resources and legacy applications.

The logical tiers are typically distributed across three independent systems:

- Client components running on local workstations (tier one)
- Application servers running on a remote server (tier two)
- Database servers running at the backend on a remote server (tier three)

However these tiers are logical and might or might not be running on the same physical system.

Tier one is responsible for the presentation and user interaction with the second-tier processes. The client components enable the user to interact in a secure manner, for example by using a secure network communication to access the second tier applications. The tier one client processes never access any tier three services on the database server directly.

In the SUT, tier one is a IBM System x® client system running a workload driver accessing sample online transaction data provided by the application server.

Tier two processes are also know as application logic layer. These processes manage the business logic of the application and provide access to the tier three services. It is the layer where most of the processing work occurs. The benchmark application used for this test emulates a Online Stock Trading System.

Tier three services are protected from direct access by the client components. Usually they are residing in a secure network (for the SUT inside the IBM System z system). Interaction is only possible through the second-tier processes.

For more information, refer to the WAS Version 8 information center:
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

Websphere Application Server (Distributed operating systems), Version 8.0 > Product overview and quick start > Product architecture > Three-tier architectures

Scenario 1: IBM WebSphere Application Server with internal HTTP transport

Figure 1 shows the System Under Test (SUT) system and software setup configured for IBM WebSphere Application Server (WAS) internal HTTP transport.

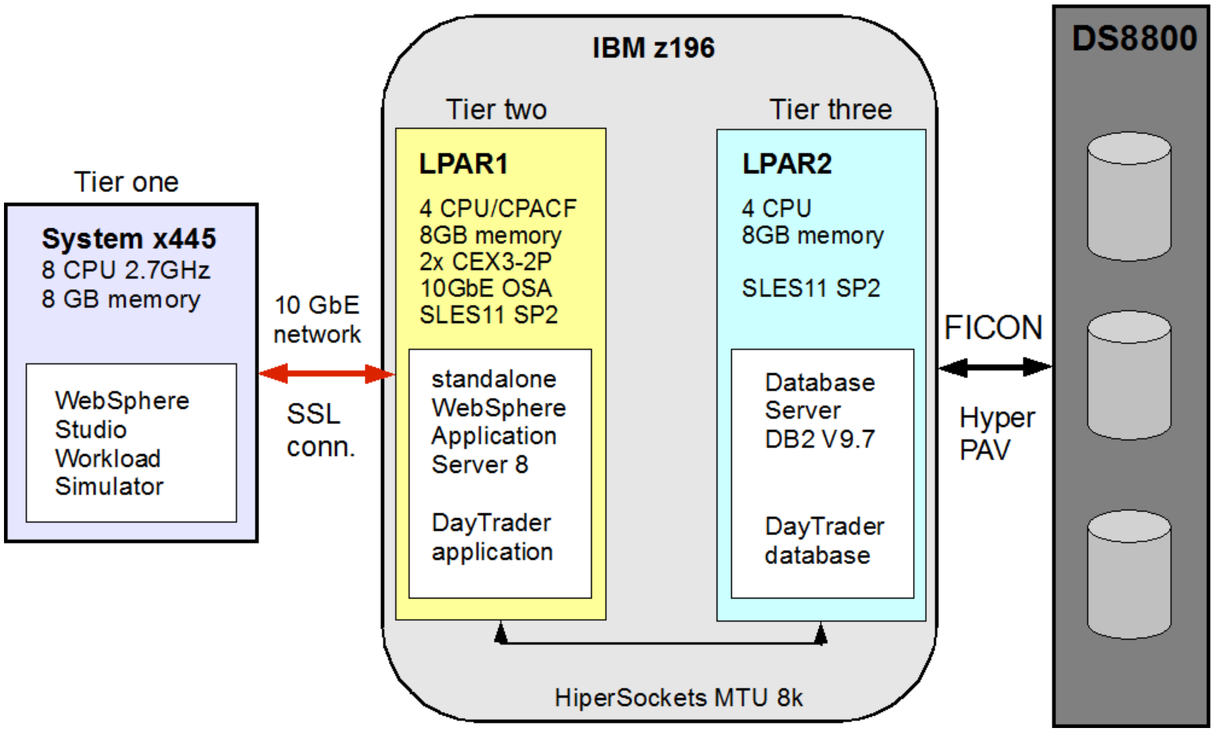


Figure 1. Scenario with WAS internal HTTP transport

WAS is installed in its own LPAR and a standalone application server profile was created. In this scenario, WAS uses its internal HTTP transport chain for communication.

The WAS LPAR is equipped with two Crypto Express3 (CEX3) features, each with two adapters. So four CEX3 adapters are available in total (however only two are used in parallel for this study).

The CP Assist for Cryptographic Function (CPACF) features must be activated using a no-charge enablement feature.

The 10GbE OSA feature is used for the communication with the client machine. This is the network connection – (2x blank) that will be SSL secured later on. Cryptographic functions required for establishing and running the SSL connection are performed on the WAS.

The WAS LPAR hosts the J2EE benchmark application (DayTrader).

The third tier in this three-tier architecture is the DB2 database server running in its own LPAR. The second-tier and the third-tier LPARs are connected through a fast internal network connection (System z HiperSockets). It is a requirement that the LPAR-to-LPAR communication method is not a bottleneck for the SUT.

The IBM System Storage DS8800 hosts the disks (DASDs) for the application and the database server. Disk attachment is based on eight FICON Express8 channels at 8 Gbps link speed. In addition Linux System z HyperPAV support is enabled to enhance the available bandwidth for the attached DASDs. The LPAR disk bandwidth must not be a bottleneck for this SUT.

The client system, representing tier one, runs a workload simulator that emulates users interacting with the application logic layer. The network connection between the client and the application server is SSL secured. How fast the application server with its resources can drive the SSL connection determines the performance of the SUT.

Scenario 2: IBM WebSphere Application Server with IBM HTTP server transport

Figure 2 shows the System Under Test (SUT) system and software setup with IBM HTTP server (HIS) acting as HTTP transport for IBM WebSphere Application Server (WAS).

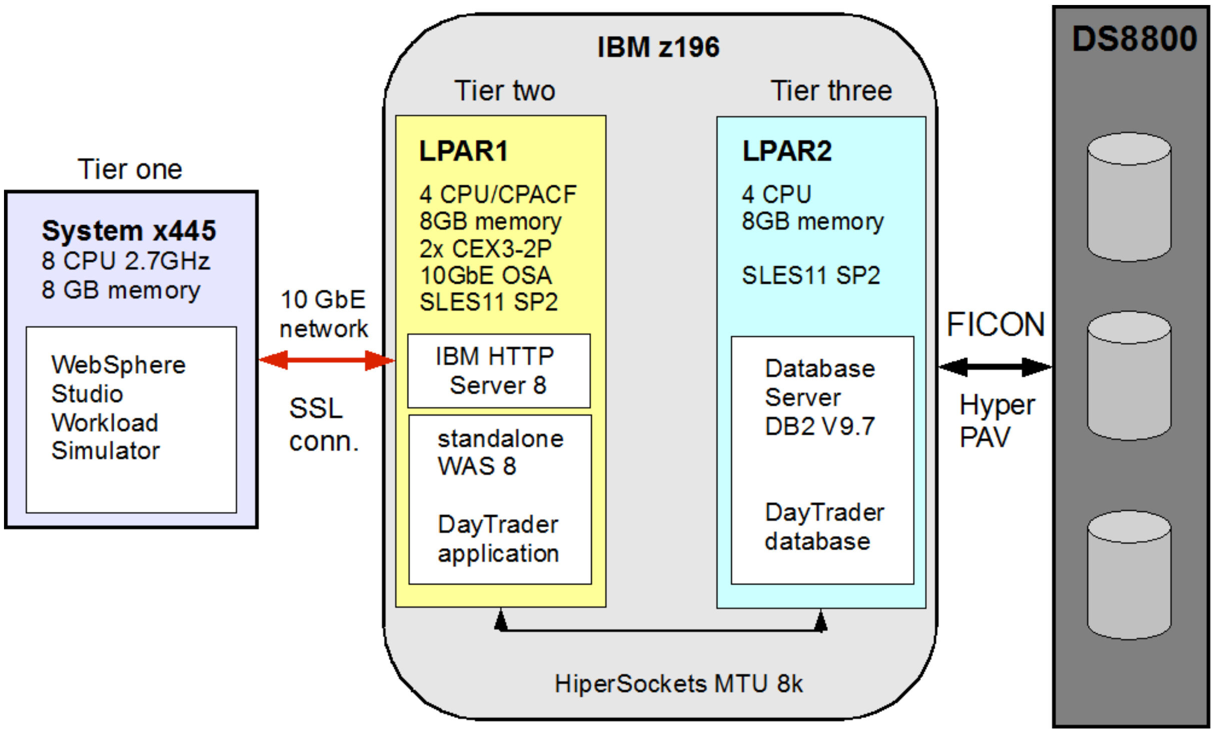


Figure 2. Scenario with WAS and IHS

Basically, this second scenario is set up in the same way as the previous scenario. It implements also a three-tier architecture, consisting of a client, an application server and a database server. The only difference is that WAS is not responsible for HTTP transport. The HTTP transport is now offloaded to the IHS. However the additional web server processes are running in the same Linux server as the WAS.

Benchmark application description

For benchmarking the DayTrader Open Source benchmark application is used.

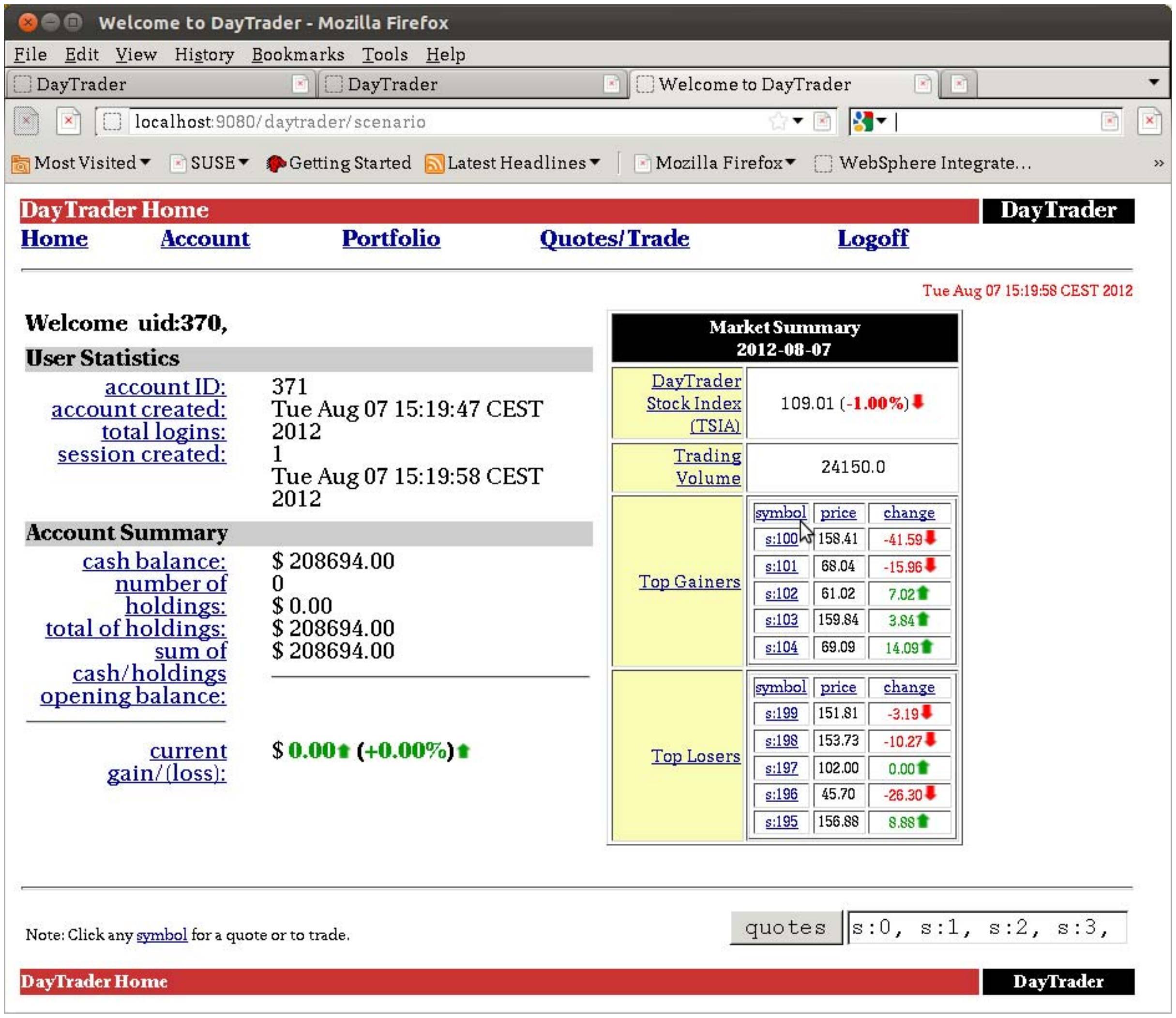
DayTrader

DayTrader is an Open Source benchmark application emulating an online stock trading system.

Users can log in and view their accounts or portfolios. In the **Quotes/Trade** section they can buy or sell stock shares out of their portfolio.

The users are emulated by a workload driver and generate load to the benchmark application. You can display the related benchmark website by accessing the DayTrader homepage. Navigate to the **Configuration Tab > Test DayTrader Scenario**.

Note: The DayTrader application must be initialized to view this page. Hence the DayTrader database must already exist and is populated with sample data. After it has been populated with sample data the DayTrader database has a size of approximately 100 MB.



DayTrader Home DayTrader

Home Account Portfolio Quotes/Trade Logoff

Tue Aug 07 15:19:58 CEST 2012

Welcome uid:370,

User Statistics

account ID:	371
account created:	Tue Aug 07 15:19:47 CEST 2012
total logins:	1
session created:	Tue Aug 07 15:19:58 CEST 2012

Account Summary

cash balance:	$ 208694.00
number of holdings:	0
total of holdings:	$ 0.00
sum of cash/holdings	$ 208694.00
opening balance:	$ 208694.00
current gain/(loss):	$ 0.00 (+0.00%)

Market Summary 2012-08-07

DayTrader Stock Index (TSIA)	109.01 (-1.00%)
Trading Volume	24150.0

Top Gainers

symbol	price	change
s:100	158.41	-41.59
s:101	68.04	-15.96
s:102	61.02	7.02
s:103	159.84	3.84
s:104	69.09	14.09

Top Losers

symbol	price	change
s:199	151.81	-3.19
s:198	153.73	-10.27
s:197	102.00	0.00
s:196	45.70	-26.30
s:195	156.88	8.88

Note: Click any symbol for a quote or to trade.

quotes s:0, s:1, s:2, s:3,

DayTrader Home DayTrader

Figure 3. DayTrader test scenario

DayTrader is currently hosted at the Apache Geronimo project. It was originally developed by IBM as the WebSphere Trade performance benchmark and was donated to the Apache Geronimo project in 2005.

DayTrader is an end-to-end Java Enterprise Edition (J2EE) web application composed of several Java classes, Java Servlets, Java Server Pages, Web Services and Enterprise Java Beans (EJB). Thus makes it an ideal benchmark application for measuring the scalability and performance of a J2EE application server like IBM WebSphere Application Server (WAS). The presentation layer is based on Java Servlets and Java Server Pages, whereas the backend business logic uses Java database connectivity (JDBC), Java Message Service (JMS), EJB and Message Bean techniques.

For further details, see the DayTrader documentation at:
https://cwiki.apache.org/GMOxDOC30/daytrader-a-more-complex-application.html

The TradeDatabase is hosted on the DB2 database server and is accessed via JDBC from the application server. In order to do a benchmark application run, the following steps are required:

1. Create TradeDatabase tables and indexes.
2. Reset DayTrader runtime to a clean starting point before each run.
3. (Re)Populate TradeDatabase with user and stock sample data.

IBM WebSphere Studio Workload Simulator

Tier one in this test scenario represents the presentation and user interaction layer.

Generally, it runs on client systems and accesses the application layer over a network connection. Typically such a network connection is SSL secured in production environments. In the case of DayTrader you require a client workload driver to generate some load.

The client application used for the System Under Test (SUT) is the IBM Websphere Studio Workload Simulator. It is a command line-based IBM internal application that simplifies the automation of test cases.

There are also a couple of Open Source developed solutions available such as, Apache JMeter, The Grinder or OpenSTA, which also would meet the requirements.

Sample output: IBM WebSphere Studio Workload Simulator

```
# clients 25/25 | rd 13.7728 MB/s | wr 0.6585 MB/s | pg 1009415 | web err 0
| resp avg = 0.006 | pg elem/s = 1673.833
IWL0058I Time limit of 600 seconds has been reached.  Shutting down.
IWL0038I Run time = 00:10:05
IWL0007I Clients completed = 0/25
IWL0059I Page elements = 1009441
IWL0060I Page element throughput = 1668.327 /s
IWL0059I Transactions = 836071
IWL0060I Transaction throughput = 1381.794 /s
IWL0059I Network I/O errors = 0
IWL0059I Web server errors = 0
IWL0059I Num of pages retrieved = 1009416
IWL0060I Page throughput = 1668.285 /s
IWL0060I HTTP data read = 8350.907 MB
IWL0060I HTTP data written = 396.831 MB
IWL0060I HTTP avg. page element response time = 0.006
IWL0059I HTTP avg. page element response time = 0 (with all clients concurrently running)
```

Based on such a report the performance of the SUT can evaluated and compared against other test scenarios or runs with different IBM WebSphere Application Server (WAS) parameters.

Several runs were done for test scenarios involving different System z cryptographic features. The results can be easily compared to each other based on the transaction numbers reported.

Preparing the DB2 database server

As outlined in the System Under Test (SUT) overview, the DB2 database server is set up in its own LPAR.

IBM DB2 Enterprise Server Edition Version 9.7 was used on the database server.

Note: The configuration is tailored for a Linux server with 8 CPUs and 8 GB memory. The database server is set up so that it does not become a performance bottleneck for the SUT. The database server tuning steps are outlined in more detail on the next pages.

Installation requirements

The IBM DB2 documentation describes the requirements for a Linux DB2 installation.

For example, the `libaio` package is required for DB2 database servers using asynchronous I/O.

See also information about the installation requirements for DB2 servers and IBM data server clients (Linux)
http://pic.dhe.**ibm.com**/infocenter/db2luw/v9r7

Database fundamentals > Installing > Database systems > Linux and UNIX > DB2 Servers > Installation prerequisites (Linux and UNIX)

Kernel parameter requirements

Starting with IBM DB2 Enterprise Server Edition Version 9.7 Fix Pack 2, the database manager uses a new formula for automatic kernel parameter adjustments.

For earlier fix pack versions, you must manually update the kernel parameter settings to meet the requirements.

DB2 database server network setup

A fast LPAR-to-LPAR connection is established between the application server and the database server.

IBM System z mainframes offer a reliable and fast internal LPAR-to-LPAR connection method called HiperSockets. HiperSockets simulate a QDIO network adapter and provide a high-speed TCP/IP communication.

The default HiperSocket frame size of 16 KB sets a TCP/IP MTU size of 8192. If required, the HiperSockets frame size can be changed in the IBM System z I/O Control Data Set (IOCDS).

The default HiperSockets frame size fits for the System Under Test (SUT), because only small or medium size network packages are expected for the benchmark application.

For details about the IOCDS, see:

Input/Output Configuration Program User's Guide, SB10-7037-10
http://www.**ibm.com**/servers/resourcelink (registration required)

For details about HiperSockets see:

HiperSockets Implementation Guide, SG24-6816
http://www.redbooks.**ibm.com**/abstracts/sg246816.html

Tuning I/O to a single DASD volume

The database server is installed on a Model 27 DASD.

This is sufficient because the TradeDatabase has a size of only a few 100 MB. In addition, the application benchmark generates only moderate database I/O, which means that an I/O-optimized logical volume setup is not required for the database log and the database data. However, the disk is hosted on an IBM storage server that has storage pool striping enabled. This provides an enlarged I/O capacity from more than a single storage server rank.

For more details, see:
http://www.**ibm.com**/developerworks/linux/linux390/perf/tuning_diskio.html#sps

However I/O throughput to a single DASD can be improved by using Parallel Access Volumes (PAV) or HyperPAV. The Linux DASD device driver can use this IBM System Storage feature to perform multiple concurrent data transfer operations to or from the same DASD volume.

To enable HyperPAV, you must create base and alias volumes, that require special IOCDS definitions. When using HyperPAV on an IBM System Storage subsystem, the alias devices are not exclusively referenced to a certain base device, but are eligible for all base devices in the same logical control unit (LCU).

The HyperPAV alias devices are managed in Linux in the same manner as a DASD base devices by using the `chccwdev` command or defining the appropriate udev rules for them.

Sample command: `lsdasd` showing four HyperPAV aliases

```
# lsdasd -s
Bus-ID     Status      Name      Device  Type  BlkSz  Size      Blocks
==============================================================================
71fa       alias                 ECKD
71fb       alias                 ECKD
71fc       alias                 ECKD
71fd       alias                 ECKD
7111       active      dasda     94:0    ECKD  4096   21129MB   5409180
```

Note: FCP channel-attached SCSI disks are not classified as DASDs and do not support HyperPAV.

For details about the IOCDS, see:

Input/Output Configuration Program User's Guide, SB10-7037-10
http://www.**ibm.com**/servers/resourcelink (registration required)

For more information about PAV and HyperPAV, see:
How to Improve Performance with PAV, SC33-8414
http://www.**ibm.com**/developerworks/linux/linux390/development_documentation.html

Setting up and tuning DB2

Some DB2 variables are configured to meet the requirements of the test setup.

Setting DB2 profile variables

These two DB2 profile variables described in this topic are explicitly set for this test setup.

- `db2set DB2_USE_ALTERNATE_PAGE_CLEANING=YES`

This variable specifies whether a DB2 database uses the alternate method of page cleaning algorithms or the default method of page cleaning. When this variable is set to `ON`, the DB2 system writes changed pages to disk, keeping ahead of `LSN_GAP` and proactively finding victims. Doing this allows the page cleaners to make better use of available disk I/O bandwidth. When this variable is set to `ON`, the `chngpgs_thresh` database configuration parameter is no longer relevant because it does not control page cleaner activity.

- `db2set DB2_PINNED_BP=YES`

Setting this variable to *YES* causes DB2 to request that Linux pins DB2 database shared memory. When you configure DB2 to pin database shared memory, make sure that the system is not over committing memory, otherwise Linux have reduced flexibility when it manages memory.

Using the DB2 autoconfigure command to configure database parameters

Using DB2 `autoconfigure` is an easy and quick way to find the initial settings for database parameters.

It calculates values for the buffer pool size, database configuration and database manager configuration parameters, with the option of suggesting or applying these recommended values.

Database administrators can use the recommended values as a basis for fine tuning the parameters.

Sample command: DB2 `autoconfigure`

```
db2 autoconfigure using
   mem_percent 80
   workload_type simple
   num_stmts 60
   tpm 10000
   is_populated yes
   num_local_apps 0
   num_remote_apps 100
   isolation rs
   bp_resizeable yes
apply db and dbm
```

where:
mem_percent 80
The TradeDatabase is the only database for this DB2 instance. Set the maximum usable instance memory to 80%.
workload_type simple
Simple workloads tend to be I/O intensive and mostly transactions.
num_stmts 60
Estimated number of statements per unit of work.
tpm 10000
Expected transactions per minute.
is_populated yes
Database populated with data
num_local_apps 0
No local applications
num_remote_apps 100
Estimated number of connected remote applications
bp_resizeable yes
Buffer pools are resizeable

Note: The command is split up in multiple lines for better readability. When issuing it at the command prompt, it needs to specified in one line.

Note: The database server is installed on a single DASD Model 27 (approximately 20 GB capacity). A more static and smaller log file setup is chosen, than that suggested by the DB2 `autoconfigure` command.

Limit the database log files to 6 GB and configure no secondary log files for the benchmark application (*LOGFILSIZ* value is 4 KB pages):

Sample commands: DB2 command sequence for changing the database log file setup.

```
db2 -v "update db cfg for tradedb using LOGPRIMARY 6"
db2 -v "update db cfg for tradedb using LOGSECOND 0"
db2 -v "update db cfg for tradedb using LOGFILSIZ 262144"
```

Setting the DB2 buffer pool size

The database server is equipped with 8 GB memory. This allows a larger buffer pool than the initial default buffer pool size.

The buffer pool size is set to 4 GB for the System Under Test (SUT). The value is in 4 KB pages.

Sample command: change DB2 default buffer pool size

```
db2 -v "alter bufferpool ibmdefaultbp size 1048576
```

Note: If the profile variable *DB2_PINNED_BP* is set, it is important to understand that the database server memory must not be overcommitted. This will result in poor performance, because the Linux memory management is capped when it runs out of memory.

Preparing the IBM WebSphere Application Server Version 8

This topic describes how the IBM WebSphere Application Server (WAS) is set up for the System Under Test (SUT).

The focus is especially on the Linux for System z cryptographic setup and the WAS cryptographic setup for securing communications using SSL.

IBM WebSphere Application Server (WAS) is running in its own LPAR and is a WAS single-system installation for the System Under Test (SUT) scenario 1.

The installation is basically the same for scenario 2, but in addition an IBM HTTP server (IHS) web server is installed.

The installed WAS software packages for both scenarios are:

- IBM Installation Manager 1.5.2
- IBM WebSphere Application Server 8.0.0.2
- Web Server Plug-ins for IBM WebSphere Application Server 8.0.0.2
- WebSphere Customization Toolbox 8.0.0.2
- IBM HTTP Server for WebSphere Application Server 8.0.0

IBM WebSphere Application Server installation requirements

To meet the IBM WebSphere Application Server (WAS) installation requirements, several packages were installed as specified in the WAS documentation.

For more information, refer to the WAS Version 8 information center:
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

Websphere Application Server (Distributed operating systems), Version 8.0 > Installing and configuring your application serving environment > Distributed operating systems > Preparing the operating system for product installation > Preparing Linux systems for installation

In addition, a web browser (for example, Mozilla Firefox) is required for the WAS administration console after the installation is complete.

Tip: One possible way to export GUIs from applications running on the WAS server to other desktops is to install the `xorg-x11-Xvnc` package on the WAS or the database server. You can log in to the servers using X11 forwarding for example, `ssh -X user@server`. Any GUIs are exported to the local X Server afterwards.

Linux kernel parameter requirements

To avoid addNode and importWasprofile problems, the Linux `ulimit` default number of open files needs to be increased.

Use the `ulimit -n` command to check the current limit. If required, change the `ulimit` number for open files. The appropriate `ulimit` command (for example, `ulimit -n 8192`) can be added to the shell profile scripts (for example, `.bashrc` or `/etc/profile.local`).

The open files limit requirement for IBM WebSphere Application Server (WAS) Version 8.0 is currently at least 8192.

For further details or if any other kernel parameters need to be changed, see:

WAS Version 8 online information center:
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

Preparing Linux systems for installation.

IBM WebSphere Application Server Network setup

The IBM WebSphere Application Server (WAS) server and database server LPAR-to-LPAR communication is set up with IBM System z HiperSockets.

See also DB2 database server network setup. The HiperSocket frame size is set to the same size as for the database server, which allows a TCP/IP MTU size of 8192 for the HiperSockets connection.

Sample command: `lsqeth` for the HiperSockets device:

```
# lsqeth hsi0
Device name                     : hsi0
---------------------------------------------
        card_type               : HiperSockets
        cdev0                   : 0.0.8000
        cdev1                   : 0.0.8001
        cdev2                   : 0.0.8002
        chpid                   : FB
        online                  : 1
        portname                : no portname required
        portno                  : 0
        route4                  : no
        route6                  : no
        checksumming            : sw checksumming
        state                   : UP (LAN ONLINE)
        priority_queueing       : always queue 2
        fake_broadcast          : 0
        buffer_count            : 128
        layer2                  : 0
        large_send              : no
        isolation               : none
        sniffer                 : 0
```

Fine tune the HiperSockets device as follows:

- Increase the number of buffers for inbound traffic to 128 (default is 64) at `/etc/udev/rules.d/51-qeth-0.0.8000.rules`
- Add the following line:

```
ACTION=="add", SUBSYSTEM=="ccwgroup", KERNEL=="0.0.8000",
ATTR{buffer_count}="128"
```

The network connection between the client system and the application server is based on 10 GbE hardware for both sides. On the client system it is a 10 GbE network card and for the application server it is a 10 GbE OSA-Express3 feature. This allows a network connection at 10 Gigabits/second link speed.

Sample command: `lsqeth` for the OSA-Express3 device

```
# lsqeth eth1
Device name                  : eth1
---------------------------------------------
        card_type            : OSD_10GIG
        cdev0                : 0.0.2100
        cdev1                : 0.0.2101
        cdev2                : 0.0.2102
        chpid                : 80
        online               : 1
        portname             : osaport
        portno               : 0
        route4               : no
        route6               : no

       checksumming         : hw checksumming
       state                : UP (LAN ONLINE)
       priority_queueing    : always queue 2
       fake_broadcast       : 0
       buffer_count         : 128
       layer2               : 0
       large_send           : no
       isolation            : none
       sniffer              : 0
```

The following OSA device driver tuning is applied for the 10 GbE OSA device:

- Increase the number of buffers for inbound traffic to 128 (default is 64) at /etc/udev/rules.d/51-qeth-0.0.2100.rules
  Add the following line:
  ACTION=="add", SUBSYSTEM=="ccwgroup", KERNEL=="0.0.2100", ATTR{buffer_count}="128"
- Enable hardware checksumming on the OSA device at /etc/udev/rules.d/51-qeth-0.0.2100.rules
  Add the following line:
  ACTION=="add", SUBSYSTEM=="ccwgroup", KERNEL=="0.0.2100", ATTR{checksumming}= "hw_checksumming"

Note: The udev file names may differ on other systems due to other subchannel numbers.

For details about IBM Linux for System z network tuning see:
http://www.**ibm.com**/developerworks/linux/linux390/perf/tuning_networking.html

Configuring the IBM WebSphere Application Server

The DayTrader benchmark application runs are conducted with a few adapted IBM WebSphere Application Server (WAS) parameters, such as larger pool sizes (for example, ORB pool, WebContainer pool, JMS thread pool).

A more significant change is the increase in the Java heap size for the application server. The memory for the WAS server is sized large enough to allow a Java heap size which is larger than the default size. A larger Java heap size can help busy applications to fulfill specific user demands. But a Java heap sized too large for the available server memory may cause swapping out memory to disk. This can have the opposite effect and slow down the system performance.

The Java heap size can be changed for each defined application server individually in WAS server profiles using the administration console as follows:

WAS admin console path: Exemplary set the Java heap to 2 GB for the application server in the SUT
Application servers > server1 > Process definition > Java Virtual Machine

```
Initial heap size: 2048 MB
Maximum heap size: 2048 MB
```

Note: Setting the initial heap size and the maximum heap size to the same size reduces the overhead of memory allocation operations on the Java heap during application runtime. However, that should be done in production environments only, when the Java heap requirements for the application servers are well understood. For benchmark applications this was done to provide stable throughput conditions.

IBM Linux on System z cryptographic setup for the IBM WebSphere Application Server SSL support

You need to install specific Linux packages and device drivers in order for the IBM WebSphere Application Server (WAS) to exploit IBM System z cryptographic hardware features.

IBM System z characteristics

The Crypto Express features (CEX) must be configured using the IBM System z Service Element (SE) or Hardware Management Console (HMC).

In the SE **Cryptographic Configuration** dialog the CEX adapters can be configured either as Accelerator or as Coprocessor in the **Crypto Type Configuration** dialog.

The LPAR hosting the IBM WebSphere Application Server (WAS) server must be capable of using the CEX features. The CEX features can be shared among selected LPARs. To enable an LPAR to use a crypto express adapter, the LPAR must be configured to have access to specific crypto adapters. This can be done on the SE or HMC with the customizing image profiles task. The IBM CP Assist for Cryptographic Functions (CPACF) is available to the IBM System z processor units (PU) when the appropriate cryptographic feature code 3863 is installed.

More information about these assignments can be found in this paper:

Exploiting IBM System z cryptographic hardware using JSSE
http://public.dhe.**ibm.com**/software/dw/linux390/perf/ZSW03153-USEN-01.pdf

JSSE setup > Cryptographic hardware > configuration

Linux packages required

The IBM WebSphere Application Server (WAS) requires some Linux packages to be installed in order to exploit the IBM System z cryptographic hardware features.

Check whether these packages are already installed, otherwise install them out of the Linux distributors repositories.

openCryptoki packages (PKCS#11 implementation for Linux):
- openCryptoki
- openCryptoki-64bit

Sample command: query installed OpenCryptoki packages
```
# rpm -qa | grep openCryptoki
openCryptoki-2.4-0.9.1
openCryptoki-64bit-2.4-0.9.1
```

libICA packages (Interface library to the ICA device driver):
- libica

Sample command: query installed libICA packages
```
# rpm -qa | grep libica
libica-2_1_0-2.1.0-0.9.1
```

The package versions and naming scheme may vary depending on the Linux distribution or distributor version.

IBM Linux on System z zcrypt device driver

The IBM Linux on System z generic cryptographic device driver (zcrypt) is required when one or more System z Crypto Express (CEX) features are accessible in a LPAR.

Coprocessors and accelerators supported by current zcrypt versions:
- PCI Cryptographic Coprocessor (PCICC)
- PCI Cryptographic Accelerator (PCICA)
- PCI-X Cryptographic Coprocessor (PCIXCC)
- Crypto Express2 Coprocessor (CEX2C)
- Crypto Express2 Accelerator (CEX2A)
- Crypto Express3 Coprocessor (CEX3C) *
- Crypto Express3 Accelerator (CEX3A) *

* Used for the System Under Test (SUT) cryptographic scenario outlined in this paper

Usually Linux kernel device driver modules are loaded with the `modprobe` command. The following command loads the zcrypt device driver called z90crypt as a Linux kernel module.

Sample command: load the zcrypt device driver
```
# modprobe z90crypt
```

On Novell SUSE Linux Enterprise Server 11 SP2 there is also a wrapper script available, that can be used for starting and querying the status of the zcrypt device driver.

Sample command: zcrypt device driver start and status query
```
# rcz90crypt start
Loading z90crypt module              done
# rcz90crypt status
Checking for module z90crypt:        running
```

The activity and status of the zcrypt supported devices can be monitored with the `lszcrypt` command. Use the maximum verbose level for the lszcrypt command to get a more detailed output about the request count and status of the cryptographic devices:

Sample command: `lszcrypt` in verbose mode
```
# lszcrypt -VV
card00:       CEX3C  online  hwtype=9  depth=8 request_count=11
card01:       CEX3A  online  hwtype=8  depth=8 request_count=0
card02:       CEX3C  online  hwtype=9  depth=8 request_count=10
card03:       CEX3A  online  hwtype=8  depth=8 request_count=0
```

The command output above lists four CEX3 adapters (two Accelerators and two Coprocessors).

All four devices are reported as online after loading the zcrypt device driver. The devices can be selectively turned on or off using the `chzcrypt` command:

Sample command: using `chzcrypt` to set adapter 00 offline

```
# chzcrypt -d 00
# lszcrypt -V
card00: CEX3C       offline
card01: CEX3A       online
card02: CEX3C       online
card03: CEX3A       online
```

The `chkconfig` command can be used to enable the service startup at system boot time.

```
# chkconfig -s z90crypt <runlevel>
```

For more information on the zcrypt device driver see

Linux on System z - Device Drivers, Features, and Commands, SC33-8411-13
http://www.**ibm.com**/support/docview.wss?uid=pub1sc33841113

CP Assist for Cryptographic Function (CPACF) support

Applications capable of offloading cryptographic operations to CPACF use, for example, the libICA library.

The libICA library includes CPACF interfaces that allow applications to make use of CPACF. This means that the IBM WebSphere Application Server (WAS) Version 8 can exploit IBM System z cryptographic features when correctly configured.

The libICA package provides a command `icainfo` that lists the libICA supported cryptographic operations for an IBM System z system. CPACF is part of the IBM System z processor complex, hence the supported operations may vary with the IBM System z system model.

Sample command: `icainfo` command output on an IBM z196 (Model 2817-M66) system

```
# icainfo
The following CP Assist for Cryptographic Function (CPACF)
operations are supported by libica on this system:
SHA-1:        yes
SHA-256:      yes
SHA-512:      yes
DES:          yes
TDES-128:     yes
TDES-192:     yes
AES-128:      yes
AES-192:      yes
AES-256:      yes
PRNG:         yes
CCM-AES-128:  yes
CMAC-AES-128: yes
CMAC-AES-192: yes
CMAC-AES-256: yes
```

Note: The output of this command becomes important when selecting middleware application cryptographic cipher suites. For example, the WAS SSL cipher suite SSL_RSA_WITH_AES_256_CBC_SHA would be a CPACF supported suite (AES-256 and SHA). It is important to select a cipher suite that uses ciphers and hashing functions supported by CPACF.

The libICA package provides another useful command `icastats`, which shows statistics about its supported cryptographic functions. Once the application cryptographic setup is done, it can be easily checked that any cryptographic operations using the libICA library are executed in hardware or software. However applications using other interfaces than libICA (for example, IBM Crypto for C (ICC)) for communicating with IBM System z cryptographic features are not considered in this statistics.

Sample command: `icastats` command output

```
# icastats
 function | # hardware | # software
----------+------------+------------
    SHA-1 |         12 |          0
  SHA-224 |          0 |          0
  SHA-256 |          0 |          0
  SHA-384 |          0 |          0
  SHA-512 |          0 |          0
   RANDOM |          1 |         35
 MOD EXPO |          7 |          0
  RSA CRT |         62 |          0
  DES ENC |          0 |          0
  DES DEC |          0 |          0
 3DES ENC |          0 |          0
 3DES DEC |          0 |          0
  AES ENC |         94 |          0
  AES DEC |         93 |          0
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
```

The output shows that the cryptographic hardware is used for the authentication process (RSA), data encryption and decryption with cipher AES and for hashes (SHA). This indicates a correctly configured setup for cryptographic hardware support.

Starting the slot manager daemon for openCryptoki - pkcsslotd

IBM WebSphere Application Server (WAS) uses the IBM Java PKCS 11 Implementation Provider (IBMPKCS11ImplProvider) to access IBM System z cryptographic features.

The IBMPKCS11ImplProvider follows the PKCS#11 (openCryptoki) standard and the PKCS#11 main API interacts with the libICA interface library in clear key mode via the ICA token managed by the slot manager daemon.

One part of this cryptographic service-library stack is the pkcsslotd daemon. When the daemon is running, it uses a cryptographic token to manage PKCS#11 objects between PKCS#11 enabled applications.

The openCryptoki package provides a startup script for the pkcsslotd daemon.

Sample command: starting the pkcsslotd daemon

```
/etc/init.d/pkcsslotd
```

Novell SLES11 SP2 includes a wrapper script, that can be used for starting and querying the status of the daemon.

Sample command: starting the pkcsslotd daemon and query its status

```
# rcpkcsslotd start
Starting pkcsslotd daemon:usermod: `root' is primary group name. done
# rcpkcsslotd status
Checking for service pkcsslotd: running
```

Note: Before using openCryptoki for the very first time on a server the script `pkcs11_startup` must to be run. The script detects available tokens from installed shared object libraries and writes corresponding records to the `pk_config_data` file. The pkcsslotd daemon uses this information when it initializes the tokens. The script is automatically called when the daemon is started with the SLES startup script `rcpkcsslotd`.

Note: The zcrypt device driver must be loaded before you start the pkcsslotd daemon, otherwise openCryptoki does not use any cryptographic hardware features.

The `chkconfig` command can be used to enable the service startup at system boot time.

```
# chkconfig -s pkcsslotd <runlevel>
```

Configuring the PKCS#11 cryptographic ICA token

The PKCS#11 cryptographic ICA token is not initialized by default. If it has never been used by any other application, it is probably not initialized and the PKCS#11 API does not work correctly when using the ICA token.

The openCryptoki package includes a utility `pkcsconf` to display and configure the state of the tokens managed by the pkcsslotd daemon.

To initialize the cryptographic ICA token with `pkcsconf` the following steps must be completed:

Sample command: display the PKCS#11 token info

```
# pkcsconf -t
Token #0 Info:
        Label: IBM ICA  PKCS #11
        Manufacturer: IBM Corp.
        Model: IBM ICA
        Serial Number: 123
        Flags: 0x880045 (RNG|LOGIN_REQUIRED|CLOCK_ON_TOKEN|
         USER_PIN_TO_BE_CHANGED|SO_PIN_TO_BE_CHANGED)
        Sessions: 0/-2
        R/W Sessions: -1/-2
        PIN Length: 4-8
        Public Memory: 0xFFFFFFFF/0xFFFFFFFF
        Private Memory: 0xFFFFFFFF/0xFFFFFFFF
        Hardware Version: 1.0
        Firmware Version: 1.0
        Time: 13:32:27
```

In this case Token #0 is the ICA token and the token is not initialized according to its flags 0x880045. The Security Officer (SO) and the user pin are not set.

An already initialized token shows these flags:
`0x44D (RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|CLOCK_ON_TOKEN|TOKEN_INITIALIZED)`

Note: The ICA token slot number (here it is #0) may differ on other Linux distributions.

The following steps show how to initialize the cryptographic token:

1. Initialize the token:
   ```
   # pkcsconf -c 0 -I
   Enter the SO PIN: ########
   Enter a unique token label: IBMICATOK
   ```

The parameter `-c` is the token slot (0 for the ICA token in this case). The SO pin is required for token initialization (default SO pin is 87654321). Enter a unique token label afterwards. This label is required later when the token is referenced from applications.

2. Set a new SO pin:

   ```
   # pkcsconf -c 0 -P
   Enter the SO PIN: ########
   Enter the new SO PIN: ########
   Re-enter the new SO PIN: ########
   ```

3. Initialize and set the user pin:

   ```
   # pkcsconf -c 0 -u
   Enter the SO PIN: ########
   Enter the new user PIN: ########
   Re-enter the new user PIN: ########
   ```

After the user pin has been initialized, the flag *USER_PIN_TO_BE_CHANGED* is set. It is recommended to explicitly set a new user pin afterwards.

```
# pkcsconf -c 0 -p
Enter user PIN: ########
Enter the new user PIN: ########
Re-enter the new user PIN: ########
```

The fully initialized ICA token now looks like this:

Sample command: display the ICA token info after its initialization

```
# pkcsconf -t
Token #0 Info:
        Label: IBMICATOK
        Manufacturer: IBM Corp.
        Model: IBM ICA
        Serial Number: 123
        Flags: 0x44D (RNG|LOGIN_REQUIRED|USER_PIN_INITIALIZED|
            CLOCK_ON_TOKEN|TOKEN_INITIALIZED)
        Sessions: 0/-2
        R/W Sessions: -1/-2
        PIN Length: 4-8
        Public Memory: 0xFFFFFFFF/0xFFFFFFFF
        Private Memory: 0xFFFFFFFF/0xFFFFFFFF
        Hardware Version: 1.0
        Firmware Version: 1.0
        Time: 15:37:35
```

Scenario 1: IBM WebSphere Application Server with internal HTTP transport SSL setup

Prerequisite: The following configuration description assumes that the Linux cryptographic setup has been configured as described in IBM Linux on System z cryptographic setup for the IBM WebSphere Application Server SSL support.

For scenario 1, the IBM WebSphere Application Server (WAS) software stack must now be configured to exploit the System z cryptographic features. In scenario 1, the WAS internal HTTP transport will drive the SSL connection.

The following setup recommendations apply to IBM Websphere Application Server Version 8.0. Other WAS versions can have a different look for the administration console.

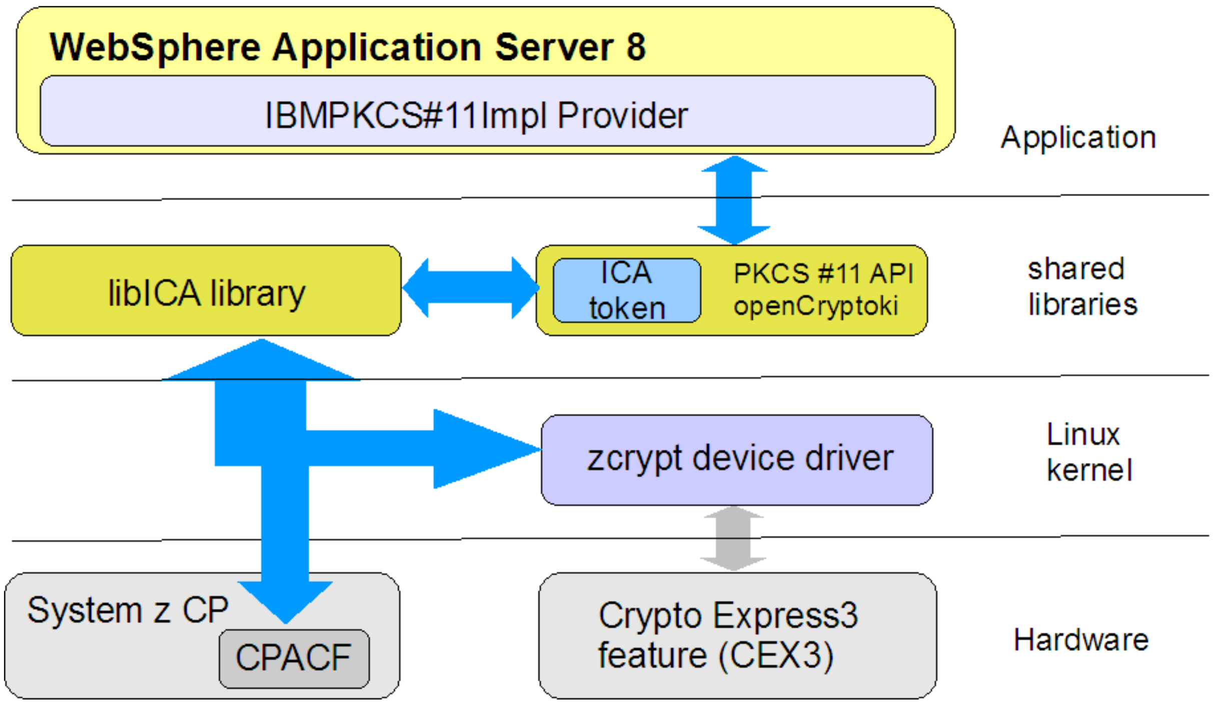


Figure 4. Overview of the cryptographic software and hardware stack for scenario 1

Figure 4 outlines the cryptographic flow showing the various levels of application interfaces, Linux shared libraries, and device drivers required to access the IBM System z cryptographic features. WAS uses the Java IBMPKCS11Impl for interaction with the PKCS#11 API (openCryptoki). openCryptoki then interacts via the ICA token with the libICA interface library to offload cryptographic operations to CP Assist for Cryptographic Function (CPACF) directly and Crypto Express3 (CEX3) using the zcrypt device driver.

Updating the Java JCE policy files

The IBM WebSphere Application Server (WAS) application stack ships its own IBM JRE environment with strong but limited jurisdiction Java Cryptography Extension (JCE) policy files.

Only limited encryption key sizes (currently 512-bit for RSA encryption) can be used by the WAS server when the JCE policy files are not upgraded to unrestricted policies. If the JCE policy files are updated, make a backup of the original jar files in case this change needs to be reverted.

For further details about JCE policy files, see:
http://www.**ibm.com**/developerworks/java/jdk/security/index.html

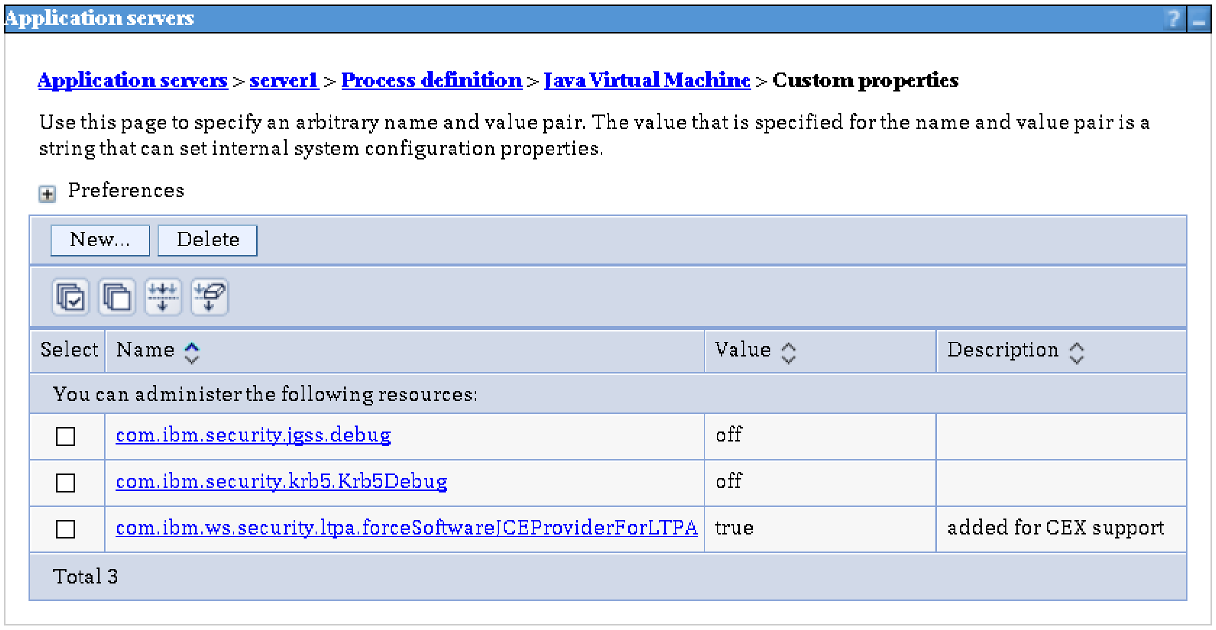
Select your Java version and search for **IBM SDK Policy files.**

Configuring the IBM WebSphere Application Server security custom property forceSoftwareJCEProviderForLTPA
This property was introduced with IBM APAR PK45677.

Add this IBM WebSphere Application Server (WAS) security custom property and set it to `true`. This prevents an "invalid library name" error when using a PKCS#11 type keystore.

To add the property using the WAS administration console, go to:

Application servers > server1 > Process definition > Java Virtual Machine > Custom properties


Application servers

Application servers > server1 > Process definition > Java Virtual Machine > Custom properties

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

Preferences

New... Delete

Select	Name	Value	Description
You can administer the following resources:			
☐	com.ibm.security.jgss.debug	off	
☐	com.ibm.security.krb5.Krb5Debug	off	
☐	com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA	true	added for CEX support
Total 3			

Figure 5. Setting the custom property for the application server

For further details about this custom property, see:

WAS Version 8 online information center under Security custom properties
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

Websphere Application Server (Distributed operating systems), Version 8.0 > Reference > Security custom properties

Configuring the IBMPKCS11Impl provider

The IBMPKCS11Impl provider maintains a Java Security properties file.

Within this file there must be a pointer to a file holding the PKCS#11 token configuration pointing to the ICA token.

For more details, see:
http://www.**ibm.com**/developerworks/java/jdk/security/

Select your IBM WebSphere Application Server (WAS) Java version. To check which Java version you have, enter the following command:
```
/opt/IBM/WebSphere/AppServer/java/jre/bin/java -version
```

Then search for the topic **PKCS 11 Implementation Provider**. This section contains a detailed description about the Java PKCS 11 Implementation Provider.

The PKCS#11 token configuration file does not follow a naming scheme, hence any filename can be chosen and needs to be specified in the Java Security properties file.

Example configuration file used for this scenario for a CEX3 feature:

```
# cat /etc/cex3config.cfg
name = IBMICATOK
description = config for IBM Crypto Express 3 (configured as an ICA token)
library = /usr/lib/pkcs11/PKCS11_API.so64
slotListIndex = 0
disabledMechanisms = {
CKM_MD5
CKM_SHA_1
CKM_MD5_HMAC
CKM_SHA_1_HMAC
CKM_SSL3_MASTER_KEY_DERIVE
CKM_SSL3_KEY_AND_MAC_DERIVE
CKM_SSL3_PRE_MASTER_KEY_GEN
}
```

where:

name
PKCS#11 (openCryptoki) cryptographic ICA token label given before
description
This string is returned by the provider instance's Provider.getInfo() method
library
Pathname of PKCS#11 implementation library
/usr/lib/pkcs11/PKCS11_API.so - 32-bit library for 32-bit JVMs
/usr/lib/pkcs11/PKCS11_API.so64 - 64-bit library for 64-bit JVMs
slotListIndex
Slot index of the PKCS#11 cryptographic ICA token
disabledMechanisms
List of PKCS#11 mechanisms to disable

Note: The PKCS#11 library path is valid for a Novell SLES11 SP2 distribution. For other distributions the PKCS#11 library path may differ.

The next step is to point from the Java Security properties file to the token configuration file. The Java Security properties file `java.security` is located in the following directory path:
`/opt/IBM/WebSphere/AppServer/java/jre/lib/security/java.security`

In the providers list section add the location of the token configuration file after the IBMPKCS11Impl provider in the list.

```
#
# List of providers and their preference orders (see above):
#
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl /etc/cex3config.cfg
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
...
```

Note:

- There must be no carriage return/line feed between the provider name IBMPKCS11Impl and the token configuration file path name! Otherwise the configuration file is not recognized.
- The IBMPKCS11Impl provider should be listed as the first provider in the list. This ensures that the IBMPKCS11Impl provider is the first one that is asked to service a specific cryptographic request.

Adding the user to the PKCS11 group

Non-root users running applications using the PKCS#11 API must belong to the pkcs11 group.

If the application server runs under a non-root user (for example, wasadmin), this user ID must be added to the pkcs11 group. The root user is automatically added when the `pkcs11_startup` command is called the first time.

List pkcs11 group members and add (optional) required user IDs:

Sample commands: add the wasadmin user to the pkcs11 group

```
# grep pkcs11 /etc/group
pkcs11:!:64:root
# usermod -G pkcs11 wasadmin
```

Selecting IBM WebSphere Application Server cipher suites

For Scenario 1 the following assumptions have been made:

- Access to the benchmark application is SSL secured with a strong encryption
- The ciphers involved should be fully supported by System z cryptographic features

The following ciphers have been chosen to conduct the performance test:

- SSL symmetric cipher: **AES-256**
- SSL asymmetric cipher: **RSA** with 2048-bit and 4096-bit key length

Note: Crypto Express3 (CEX3) feature support of RSA keys with 4096-bit length became available for z196 with MCL N29766.021 in December, 2010.

Various SSL cipher suites can be enabled or disabled using the IBM WebSphere Application Server (WAS) administration console. For the System Under Test (SUT) a single cipher suite is selected to force the use of the given ciphers.

Production systems often have other requirements related to supported SSL cipher suites for an application server. Usually they are not restricted to a single suite. However, you should take care that the selected cipher suites are fully supported by IBM System z cryptographic features. A lot of cipher suites are only partially or not supported by cryptographic hardware features.

Tip: `icainfo` lists ciphers supported by libICA. Use the `icastats` command to check that the desired ciphers show request counts in the hardware column.

Table 7 shows some examples of RSA-AES cipher suite variants offered by WAS Version 8. But not all cipher suites are supported in the same manner. First, depending on the System z system (CPACF level) and CEX features, not all ciphers are supported. Second, not all cipher suite variants are supported.

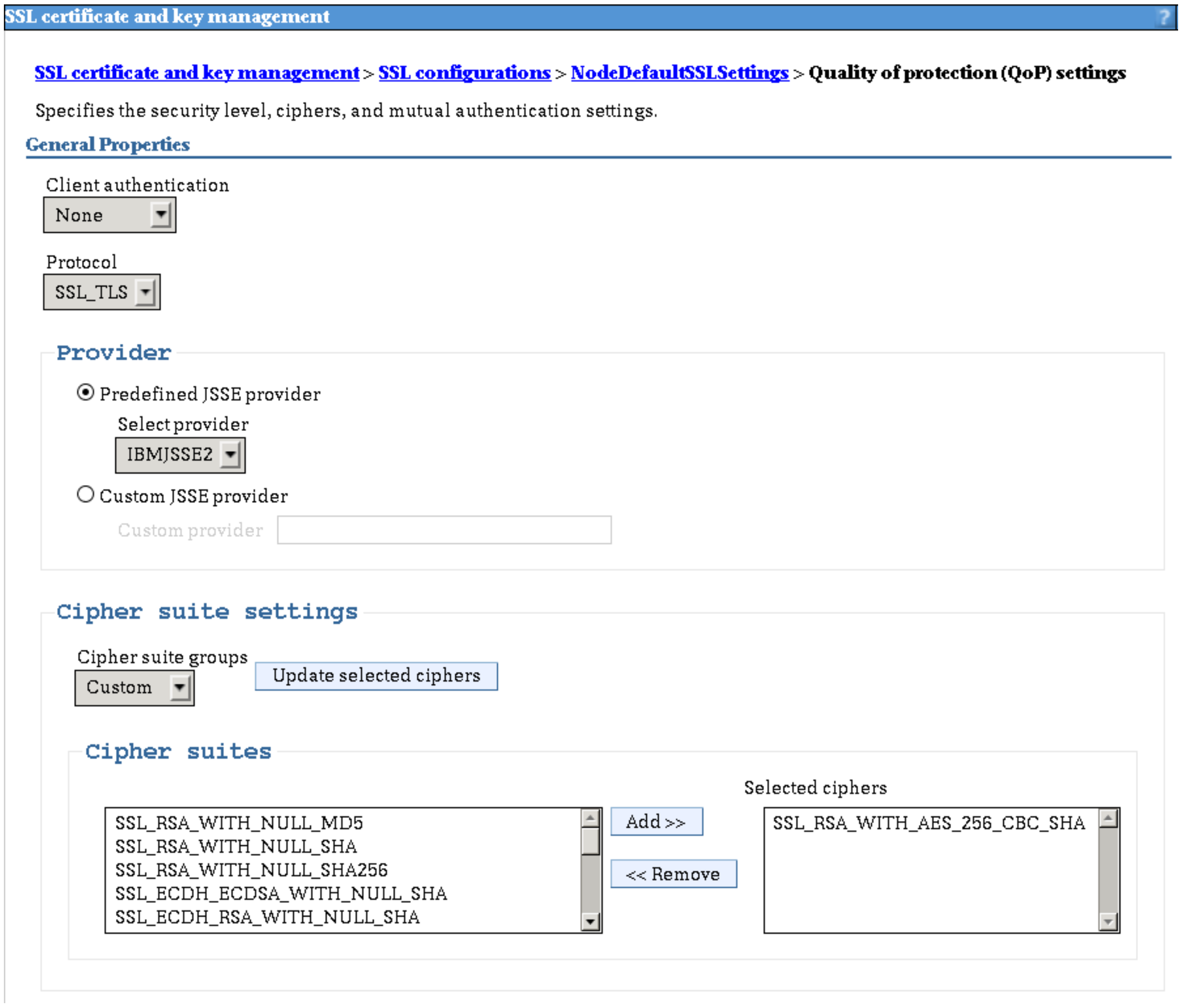
Table 7. Overview of WAS SSL cipher suites with AES-256

WAS Version 8 cipher suite	IBM System z cryptographic stack support
SSL_RSA_WITH_AES_256_CBC_SHA	Full support
SSL_RSA_WITH_AES_256_CBC_SHA256	Not supported Currently no SHA-256 support for openCryptoki / ICA token with RSA *
SSL_DHE_RSA_WITH_AES_256_CBC_SHA	Only partially supported DHE-RSA in software; AES in hardware
SSL_ECDH_RSA_WITH_AES_256_CBC_SHA	Not supported ECDH-RSA currently not supported
SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA	Not supported ECDHE-RSA currently not supported

* SHA-256 standalone is supported (see icainfo output), but not in combination with a RSA cipher suite.

The WAS administration console provides a dialog for configuring SSL cipher suites. The administration console navigation path is:

Security > SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Quality of protection (QoP) settings



SSL certificate and key management

SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings > Quality of protection (QoP) settings

Specifies the security level, ciphers, and mutual authentication settings.

General Properties

Client authentication: None

Protocol: SSL_TLS

Provider

[✓ Predefined JSSE provider]

Select provider: IBMJSSE2

Custom JSSE provider

Custom provider

Cipher suite settings

Cipher suite groups: Custom

Update selected ciphers

Cipher suites

SSL_RSA_WITH_NULL_MD5
SSL_RSA_WITH_NULL_SHA
SSL_RSA_WITH_NULL_SHA256
SSL_ECDH_ECDSA_WITH_NULL_SHA
SSL_ECDH_RSA_WITH_NULL_SHA

Add >>

<< Remove

Selected ciphers: SSL_RSA_WITH_AES_256_CBC_SHA

Figure 6. Quality of Protection (QoP) settings

SSL_RSA_WITH_AES_256_CBC_SHA has been chosen for the SUT. This suite is fully supported by the System z cryptographic features and meets the requirements for the test as a strong cipher suite.

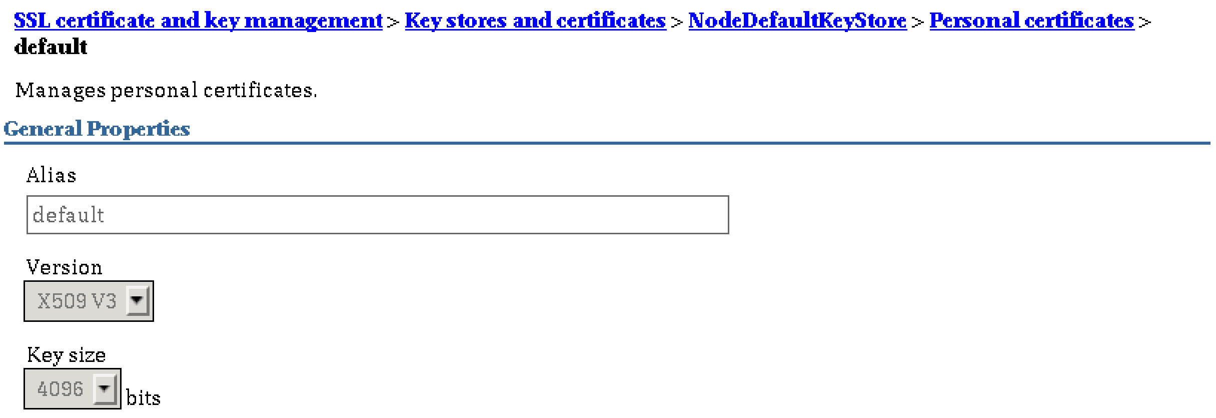
SSL certificate key sizes and key management

The IBM WebSphere Application Server (WAS) default certificates are stored in a default keystore *NodeDefaultKeyStore*.

This keystore has the type PKCS12, which is the default Java keystore type. Every stored certificate in a keystore was created with a certain key size.

The certificates in a keystore can be viewed using the administration console navigation path:

Security > SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates



SSL certificate and key management > Key stores and certificates > NodeDefaultKeyStore > Personal certificates > **default**

Manages personal certificates.

General Properties

Alias: default

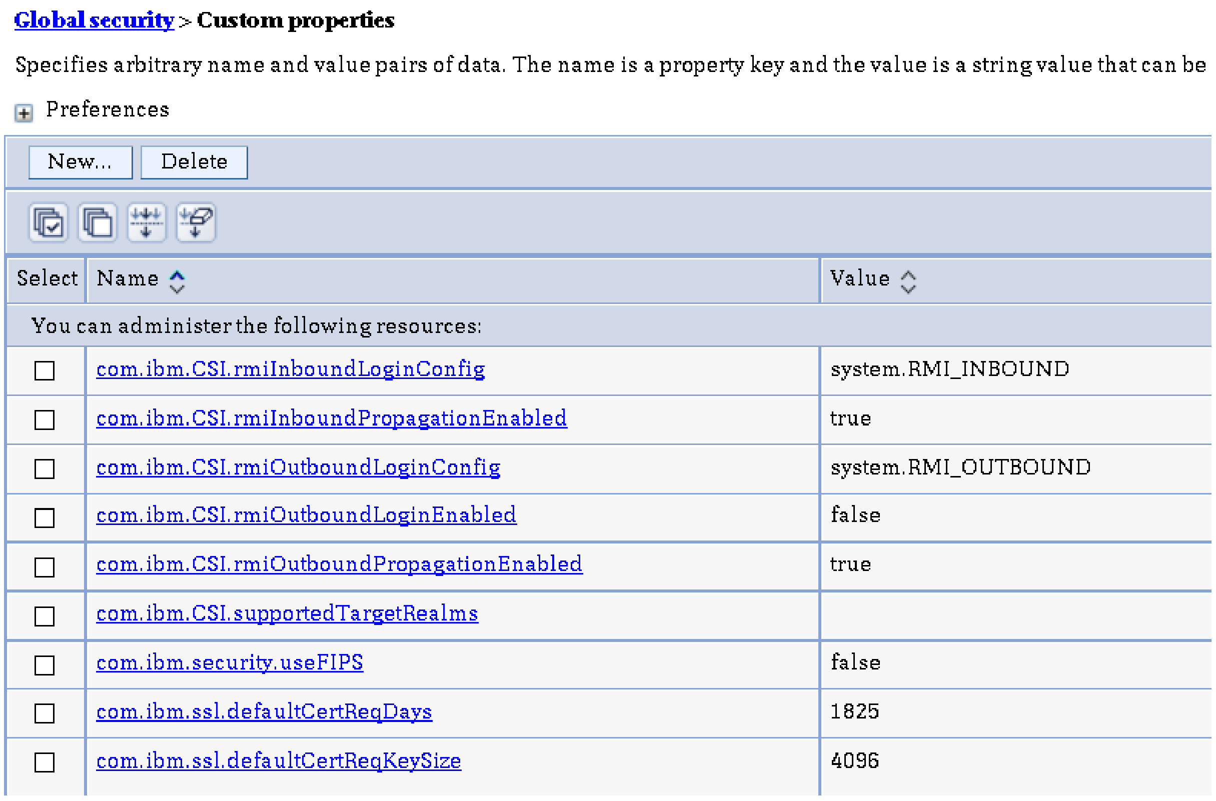
Version: X509 V3

Key size: 4096 bits

Figure 7. The general properties show also the key size for the public key in the certificate

The default certificate key size (used when creating new certificates) can be changed by adding a specific property. The administration console navigation path is:

Security > Global security > Custom properties



Global security > **Custom properties**

Specifies arbitrary name and value pairs of data. The name is a property key and the value is a string value that can be

Select	Name	Value
	You can administer the following resources:	
☐	com.ibm.CSI.rmiInboundLoginConfig	system.RMI_INBOUND
☐	com.ibm.CSI.rmiInboundPropagationEnabled	true
☐	com.ibm.CSI.rmiOutboundLoginConfig	system.RMI_OUTBOUND
☐	com.ibm.CSI.rmiOutboundLoginEnabled	false
☐	com.ibm.CSI.rmiOutboundPropagationEnabled	true
☐	com.ibm.CSI.supportedTargetRealms	
☐	com.ibm.security.useFIPS	false
☐	com.ibm.ssl.defaultCertReqDays	1825
☐	com.ibm.ssl.defaultCertReqKeySize	4096

Figure 8. Custom property for default certificate key size

The added property is `ssl.client.props:com.ibm.ssl.defaultCertReqKeySize` with the new default key size (for example, 2048-bit or 4096-bit).

Newly created certificates use the key size from this property as a default.

Checking the cryptographic setup

The next step is to check the cryptographic setup and verify that everything works as expected.

To do this, access an application on the IBM WebSphere Application Server (WAS) using SSL.

To verify the setup, access an application on the IBM WebSphere Application Server (WAS) using SSL.

For example, do a sniff test with a server data transfer tool such as `curl`. First restart the application server so that all previous modifications become active.

The following example uses the WAS default SSL port 9443 for internal HTTP transport to access the benchmark application DayTrader, which has already been installed. The TLS handshake process succeeds and the AES-256 cipher is used.

Sample command: using `curl` to access DayTrader via internal WAS HTTP port

```
# curl -k -v https://wasnode1.net:9443/daytrader
* About to connect() to wasnode1.net port 9443 (#0)
*   Trying 10.x.x.x... connected
* Connected to wasnode1.net (10.x.x.x) port 9443 (#0)
* successfully set certificate verify locations:
*   CAfile: none
  CApath: /etc/ssl/certs/
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using AES256-SHA
```

Check the output of the `lszcrypt` command. The request count for any active CEX3 features should show some requests. In this case a CEX3 accelerator was used, whereas the other cards are offline.

Sample command: `lszcrypt` showing processed requests on a CEX3A

```
# lszcrypt -VV
card00: CEX3C       offline hwtype=9  depth=8 request_count=3
card01: CEX3A       online  hwtype=8  depth=8 request_count=69
card02: CEX3C       offline hwtype=9  depth=8 request_count=0
card03: CEX3A       offline hwtype=8  depth=8 request_count=0
```

Check is the output of the `icastats` command. The ciphers from the selected WAS cipher suite should show counts in the hardware column.

Sample command: `icastats` statistics listing requests in the hardware column

```
# icastats
 function | # hardware | # software
----------+------------+------------
    SHA-1 |         12 |          0     <- SHA-1 (CPACF)
  SHA-224 |          0 |          0
  SHA-256 |          0 |          0
  SHA-384 |          0 |          0
  SHA-512 |          0 |          0
   RANDOM |          1 |         35     <- not supported with CEX3A
 MOD EXPO |          7 |          0
  RSA CRT |         62 |          0     <- RSA cipher (CEX3A)
  DES ENC |          0 |          0
  DES DEC |          0 |          0
 3DES ENC |          0 |          0
 3DES DEC |          0 |          0
  AES ENC |         94 |          0     <- AES-256 encryption (CPACF)
  AES DEC |         93 |          0     <- AES-256 decryption (CPACF)
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
```

Scenario 2: IBM WebSphere Application Server with IBM HTTP server transport SSL setup

Prerequisites: The following configuration description assumes that the Linux cryptographic setup has been configured as described in IBM Linux on System z cryptographic setup for the IBM WebSphere Application Server SSL support. The IBM WebSphere Application Server (WAS) web server plugin is already configured, so that IBM HTTP server (IHS) and WAS can communicate.

For additional information on configuring the WAS web server plugin, see

WAS Version 8 information center:
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

Websphere Application Server (Distributed operating systems), Version 8.0 > Setting up intermediary services > Implementing a web server plug-in

Based on the Linux cryptographic setup completed in IBM Linux on System z cryptographic setup for the IBM WebSphere Application Server SSL support, the ÍBM System z cryptographic features should be ready to use from an operating system point of view. For this scenario, the IHS must be configured to exploit the IBM System z cryptographic features. In this scenario IHS drives the SSL connection.

The setup recommendations in the following match IHS Version 8.0 for WAS Version 8.

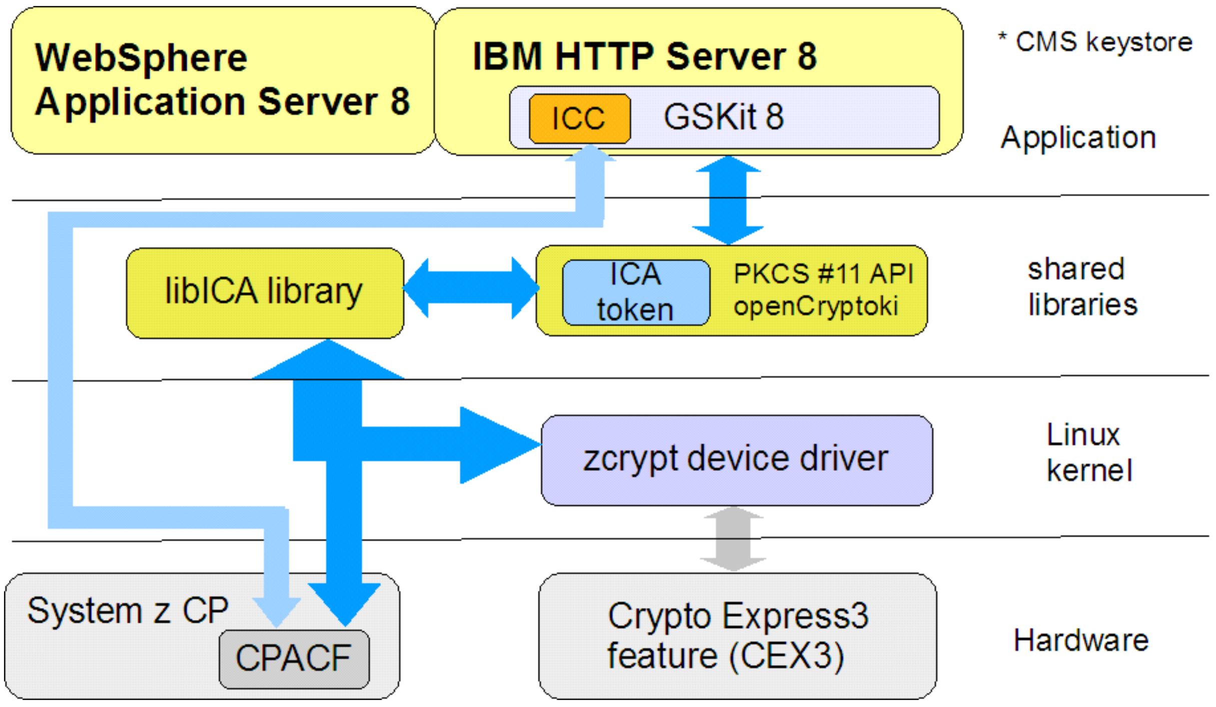


Figure 9. Overview of the cryptographic software and hardware stack for scenario 2

Figure 9 shows the cryptographic flow through the various levels of application interfaces, Linux shared libraries and device drivers to access the IBM System z cryptographic features. IHS uses the Global Secure ToolKit API (GSKit) to interact with the PKCS#11 API (openCryptoki). openCryptoki then uses the libICA interface library via the ICA token to offload cryptographic operations to CPACF directly and CEX3 over the zcrypt device driver. GSKit also uses IBM Crypto for C (ICC) for some cryptographic operations.

Attention: For previous IHS versions it was required to remove `gskikm.jar` file from the `/opt/IBM/HTTPServer/java/jre/lib/ext` directory.

Do not perform this step if you are using IHS Version 8.

IBM HTTP server certificate management

The `iKeyman` utility or the `gskcmd` command utility can be used to create a self-signed server certificate.

The certificate is used by the SSL protocol when it secures communications between clients and the application server. Self-signed server certificates can be also used when you act as your own CA for a private web network or for benchmark application testing purposes.

Example of creating a self-signed server certificate using the `gskcmd` command utility:

```
# gskcmd -cert -create -size 4096 -dn "CN=wasnode1.net,O=IBM,C=DE" -label
ihscert -crypto /usr/lib/pkcs11/PKCS11_API.so64 -tokenlabel IBMICATOK -pw XXXX
```

The command creates a self-signed server certificate `ihscert` with 4096-bit key size. It is stored in the PKCS#11 cryptographic ICA token with the label IBMICATOK. The password XXX is the user pin specified when the ICA token is initialized, see Configuring the PKCS#11 cryptographic ICA token

Display the self-signed server certificate using the `gskcmd` command utility:

```
# gskcmd -cert -details -label ihscert -crypto /usr/lib/pkcs11/PKCS11_API.so64
-tokenlabel IBMICATOK -pw XXXX
Label: ihscert
Key Size: 4096
Version: X509 V3
Serial Number: 50 3B 61 E9
Issued by: CN=wasnode1.net, O=IBM, C=DE
Subject: CN=wasnode1.net, O=IBM, C=DE
Valid: From: Monday, 27 August 2012 14:02:49 o'clock CEST
To: Tuesday, 27 August 2013 14:02:49 o'clock CEST
Fingerprint: 9D:FC:17:90:4B:59:39:52:20:D9:F6:22:3E:D1:48:4F:1A:B0:13:3D
Signature Algorithm: SHA1withRSA (1.2.840.113549.1.1.5)
Trust Status: enabled
```

The `gkscmd` command displays the self-signed certificate stored in the PKCS#11 ICA token.

IBM HTTP server (IHS) signer certificates are stored in a Certificate Management Services (CMS) key store created with the GSKit installed with IHS.

So the next step is to create a CMS keystore for the signer certificates.

The password for the keystore should be saved in a password file by using the `-stash` option. This is required for the IHS SSL configuration later on. Choose an appropriate directory location for the CMS keystore. The command generates three files.

Create a CMS keystore for signer certificates:

```
# gskcmd -keydb -create -db /opt/IBM/HTTPServer/ssl/key.kdb -pw 1234
-type cms -expire 999 -stash

# ls /opt/IBM/HTTPServer/ssl
key.kdb  key.rdb  key.sth
```

Stashing the PKCS#11 cryptographic ICA token user PIN

IBM HTTP server (IHS) requires access to the PKCS#11 ICA token for the self-signed certificate.

Therefore the token user pin needs to be stashed into a password file.

```
# sslstash -c /opt/IBM/HTTPServer/ssl/ibmicatok.sth crypto XXXX
```

Where `XXXX` is the user pin of the cryptographic ICA token. This command creates a password stash file called `ibmicatok.sth`. The file is also used in the IHS SSL configuration later on.

Adding the IBM HTTP server user to the pkcs11 group

Non-root users running applications using the PKCS#11 API must belong to the pkcs11 group.

IBM HTTP server (IHS) usually runs under the Linux nobody user. So the nobody user must be added to the pkcs11 group. The root user is automatically added when the pkcss_startup is called the first time.

List pkcs11 group members and add the nobody user if applicable:

Sample commands: adding the Linux nobody user to the pkcs11 group

```
# grep pkcs11 /etc/group
pkcs11:!:64:root
# usermod -G pkcs11 nobody
```

Configuring IBM HTTP server SSL support

IBM HTTP server (IHS) maintains a file where all its configuration is stored.

The configuration file for IHS has the following default path:

```
/opt/IBM/HTTPServer/conf/httpd.conf
```

Any SSL configurations need to go into this file. The following example SSL definition supports the Transport Security Layer (TLS) protocol and IBM System z cryptographic features, where TLS is the successor for the SSL.

```
# Example SSL(TLS) configuration
#
# required due to GSKit8 library problem
LoadFile /usr/lib64/libcrypto.so
#
LoadModule ibm_ssl_module modules/mod_ibm_ssl.so
Listen 443
<VirtualHost wasnode1.net:443>
ServerName wasnode1.net
SSLEnable
SSLProtocolDisable SSLv2
SSLProtocolDisable SSLv3
# cipher suite TLS_RSA_WITH_AES_256_CBC_SHA(35b)
# remove all ciphers first
SSLCipherSpec ALL NONE
SSLCipherSpec ALL +TLS_RSA_WITH_AES_256_CBC_SHA
# symmetric offload (older GSKit versions)
# SSLAttributeSet 417 549
</VirtualHost>
# PKCS#11 configuration
KeyFile /opt/IBM/HTTPServer/ssl/key.kdb
SSLServerCert IBMICATOK:ihscert
SSLStashfile /opt/IBM/HTTPServer/ssl/ibmicatok.sth
SSLPKCSDriver /usr/lib/pkcs11/PKCS11_API.so64
SSLDisable
SSLCachePortFilename /opt/IBM/HTTPServer/logs/siddport
# End of SSL configuration
```

For more details on IHS SSL directives see IBM WebSphere Application Server (WAS) Version 8 information center:
http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0/index.jsp

IBM HTTP Server for Websphere Application Server, Version 8.0 > Reference > Settings > SSL directives

Explanation of some SSL httpd.conf directives

LoadFile /usr/lib64/libcrypto.so

This statement needs to be added due to conflicting GSKit8 and openSSL libraries. The problem currently occurs on a Novell SLES11 SP2 distribution with WAS8 (GSKit 8). It has also been observed for older distributions and IHS GSKit versions (for example using GSKit 7.0.4.14). Later Novell SLES or Red Hat distributions may not require this circumvention.

The following circumventions have been tested:

(a) Put the following statement in the `httpd.conf` file:

```
LoadFile /usr/lib64/libcrypto.so
```

(a1) As an alternative to the preferred step (a), add the following to the IBM HTTP Server `bin/envvars` file:

```
LD_PRELOAD=/usr/lib/libcrypto.so:/usr/lib64/libcrypto.so
export LD_PRELOAD
```

(b) For the iKeyman utility, add the following at the top of the IHS `bin/ikeyman_startup` script:

```
LD_PRELOAD=/usr/lib/libcrypto.so:/usr/lib64/libcrypto.so
```

Error indication: IHS does not start with SSL support enabled. The `IHS error_log` file shows:

```
[Mon Aug 20 14:46:52 2012] [crit] Error 430 initializing SSL environment, aborting
startup

[Mon Aug 20 14:46:52 2012] [error] SSL0153E: Initialization error,
The PKCS#11 driver failed to find the token specified by the caller.
```

When IHS starts successfully with SSL enabled the `error_log` reports:

```
[Mon Aug 27 17:05:16 2012] [notice] Using GSKit version 8.0.14.9
```

See also the IBM support technote:
http://www.**ibm.com**/support/docview.wss?uid=swg21313367

SSLProtocolDisable SSLv2, SSLProtocolDisable SSLv3

Disabling SSL protocols SSLv2 and SSLv3 to force TLS in this example.

SSLCipherSpec ALL NONE

SSLCipherSpec ALL +TLS_RSA_WITH_AES_256_CBC_SHA

An example for the long name cipher suite specification. `ALL NONE` removes all default ciphers from the list. Then any desired cipher suites can be added to the list, indicated by the prefix plus sign. Only one cipher suite is added here to force the usage of the selected ciphers for the benchmark application. See also the comments in Selecting IBM WebSphere Application Server cipher suites.

# SSLAttributeSet 417 549

Not used here. This was required for older GSKit versions, so that CPACF can be used for the symmetric cipher offload. GSKit8 no longer requires this directive. However, verify with the `icastats` command that the chosen symmetric ciphers (3DES, AES) are displayed in the hardware column.

SSLServerCert IBMICATOK:ihscert

Sets the server certificate to use for this virtual host. In this case it is the PKCS#11 cryptographic ICA token that holds the server certificates. The token label must be the same as the label given during the PKCS#11 token initialization process. The name of the server certificate follows after the token.

SSLStashfile /opt/IBM/HTTPServer/ssl/ibmicatok.sth

Name of the password file that holds the stashed PKCS#11 cryptographic ICA token user PIN.

SSLPKCSDriver /usr/lib/pkcs11/PKCS11_API.so64

Fully qualified name of the PKCS#11 library module.

Checking the cryptographic setup

The next step is to check the cryptographic setup and verify that everything works as expected.

To do this, access an application on the IBM HTTP server (IHS) - IBM WebSphere Application Server (WAS) combination using SSL.

For example, do a sniff test with a server data transfer tool such as `curl`. Restart IHS and the application server so that all previous modifications become active.

The following example uses the IHS default SSL port 443 to access the benchmark application DayTrader, which has already been installed. The TLS handshake process succeeds and the AES-256 cipher is used.

Sample command: using `curl` for accessing DayTrader via internal IHS standard SSL port

```
# curl -k -v https://wasnode1.net:443/daytrader
* About to connect() to wasnode1.net port 443 (#0)
*   Trying 10.x.x.x... connected
* Connected to wasnode1.net (10.x.x.x) port 443 (#0)
* successfully set certificate verify locations:
*   CAfile: none
  CApath: /etc/ssl/certs/
* SSLv3, TLS handshake, Client hello (1):
* SSLv3, TLS handshake, Server hello (2):
* SSLv3, TLS handshake, CERT (11):
* SSLv3, TLS handshake, Server finished (14):
* SSLv3, TLS handshake, Client key exchange (16):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSLv3, TLS change cipher, Client hello (1):
* SSLv3, TLS handshake, Finished (20):
* SSL connection using AES256-SHA
...
```

Check the output of the `lszcrypt` command. The request count for any active Crypto Express3 (CEX3) features now shows some requests. In this case a CEX3 Coprocessor, whereas the other cards are offline.

Sample output: `lszcrypt` showing processed requests on a CEX3C

```
# lszcrypt -VV
card00: CEX3C       online  hwtype=9  depth=8 request_count=100
card01: CEX3A       offline hwtype=8  depth=8 request_count=0
card02: CEX3C       offline hwtype=9  depth=8 request_count=19
card03: CEX3A       offline hwtype=8  depth=8 request_count=0
```

Check the output of the `icastats` command. The ciphers from the selected WAS cipher suite display counts in the hardware column.

Sample command: `icastats` statistics showing requests in the hardware column

```
# icastats
 function | # hardware | # software
----------+------------+------------
    SHA-1 |        120 |          0     <- SHA-1 (CPACF)
  SHA-224 |          0 |          0
  SHA-256 |          0 |          0
  SHA-384 |          0 |          0
  SHA-512 |          0 |          0
   RANDOM |        211 |          0     <- RANDOM functions (CEX3C)
 MOD EXPO |         68 |          0
  RSA CRT |         10 |          0     <- RSA (CEX3C)
  DES ENC |          0 |          0
  DES DEC |          0 |          0
 3DES ENC |          0 |          0
 3DES DEC |          6 |          0
  AES ENC |         30 |          0     <- AES-256 encryption (CPACF)
  AES DEC |         30 |          0     <- AES-256 decryption (CPACF)
 CMAC GEN |          0 |          0
 CMAC VER |          0 |          0
```

SSL performance results

The results for the DayTrader benchmark application are discussed in this topic.

The benchmark application was executed for both scenarios outlined in this paper. Observations made for each of the scenarios are illustrated in the following.

The benchmark runs are done with different combinations of IBM System z cryptographic features. Different key sizes for the asymmetric RSA cipher are also taken into account. The RSA cipher is the most CPU cost intensive part of a SSL cipher suite. The larger the RSA key size the more cost intensive are the cipher operations. This influences the SSL performance and limits the maximum possible transaction throughput.

For a description of the DayTrader benchmark application, see DayTrader.

Table 8 describes the labels used for the cryptographic setups on the following charts.

Table 8. Description of chart labels for the cryptographic setups used

Chart label	Cryptographic setup description
Software	Software means no IBM System z cryptographic features are used to accelerate or offload cryptographic operations. All cryptographic operations are done in fallback routines of the cryptographic libraries and hence will be calculated in software.
CPACF	Only the CP Assist for Cryptographic Function (CPACF) feature is used for symmetric cipher operation acceleration, but no Crypto Express3 (CEX3) features are involved.
1-CEX3[A/C]	One CEX3 adapter is used for the cryptographic setup. The adapter is either configured as a Co-Processor (C) or as an Accelerator (A). The setups also include the CPACF feature.
2-CEX3[A/C]	Two CEX3 adapters are used for the cryptographic setup. The adapters are either configured as Co-Processors (C) or as Accelerators (A). The setups also include the CPACF feature.

Results for scenario 1: IBM WebSphere Application Server internal HTTP transport

For scenario 1 the IBM WebSphere Application Server (WAS) application server with internal HTTP transport drives the SSL connection.

This means that secure client HTTPS requests are directed to the WAS directly. The self-signed certificate in the WAS PKCS#12 keystore was defined with a key size of 2048-bit and 4096-bit in a second measurement series, where the scaling is done over the cryptographic setups.

Note: The workload generated (number of virtual users) by the workload driver is kept constant across all measured scenarios.

See chapter 4 for a detailed description of the System Under Test (SUT) scenarios.

RSA key size 2048-bit

Results for tests performed for scenario 1 using RSA key size 2048-bit are discussed in this topic.

DayTrader transaction throughput and IBM WebSphere Application Server LPAR CPU load

Figure 10 shows the normalized DayTrader SSL transaction throughput, when scaling the cryptographic setup and using a 2048-bit RSA key.

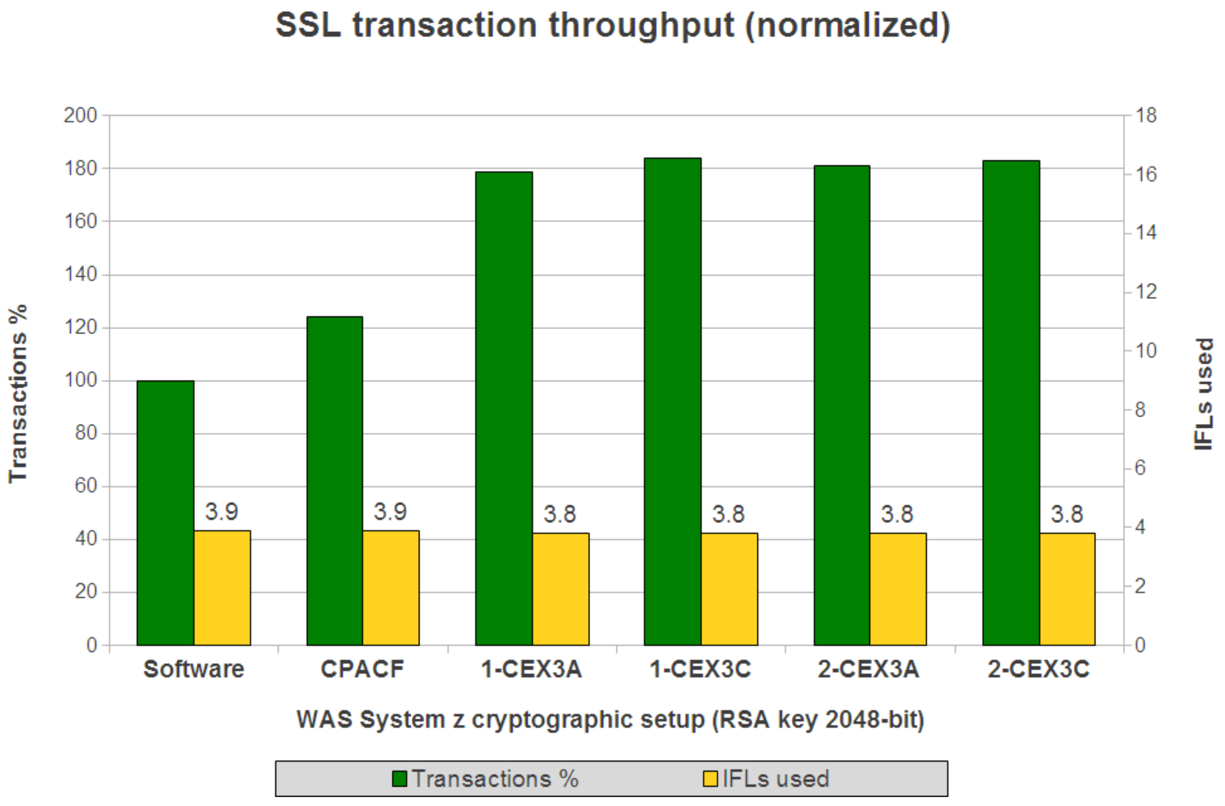


Figure 10. SSL transaction throughput when a 2048-bit RSA key is used

The green (dark) bars in Figure 10 show the DayTrader SSL transaction throughput for each of the cryptographic setups. The throughput number for the "Software" setup defines the 100% baseline for the other cryptographic setups.

Observation
The transaction throughput can be almost doubled when the IBM System z cryptographic features are used.

Conclusion
Approximately 30% of the throughput increase goes back to the CP Assist for Cryptographic Function (CPACF) and about 70% to Crypto Express3 (CEX3) for this type of workload. A second CEX3 processor does not result in higher throughput numbers, because a single CEX3 processor was sufficient to handle the workload. For the 2-CEX3 setups the zcrypt device driver is doing a load-balancing among the available CEX3 processors.

The IBM WebSphere Application Server (WAS) LPAR is equipped with 4 Integrated Facilities for Linux (IFLs).

Observation
All 4 IFLs are almost fully used on the WAS LPAR across all cryptographic setups in this scenario (see yellow/light bars).

Conclusion
For the setups with IBM System z cryptographic features, CPU time is freed due to offloading cryptographic operations to the cryptographic features. The freed CPU time can be used to handle a greater amount of transaction requests compared to 'Software' setup. This leads to a higher SSL transaction throughput number in total for the setups exploiting the IBM System z cryptographic features.

DayTrader transaction CPU costs

Another important aspect with respect to the reached throughput numbers is the amount of CPU required to drive the workload, considered as cost factor. This metric is calculated as the amount of CPU needed to drive a certain amount of transactions. Figure 11 shows the CPU costs when a 2048-bit RSA key is used.

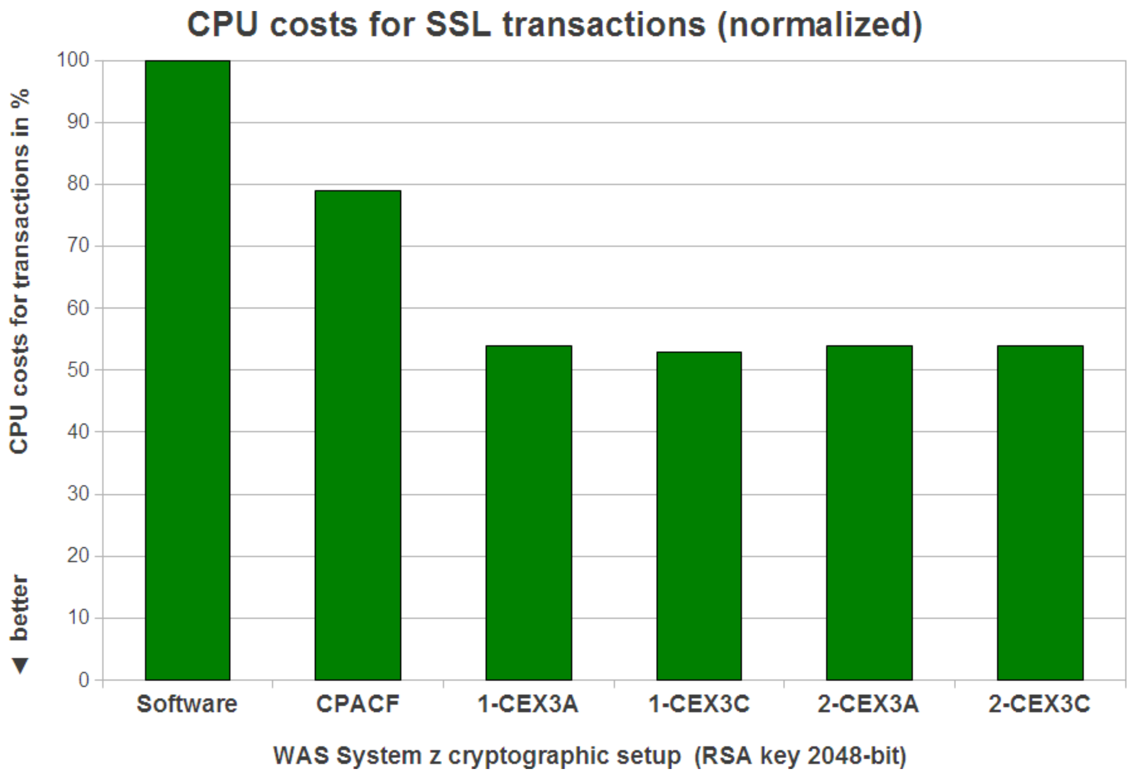


Figure 11. CPU costs for SSL transactions when a 2048-bit RSA key is used

The chart shows the CPU costs for the SSL transactions when using a 2048-bit RSA key. The numbers are normalized against the 'Software' CPU costs number.

Observation
The CPU costs for the transactions are nearly half as much for the CEX3 measurement series.

Conclusion
Exploitation of the IBM System z cryptographic features reduce the CPU costs for SSL transactions dramatically. The reduced CPU costs allow more incoming transactions to be processed and a higher transaction throughput rate is possible.

DayTrader average response times

Figure 12 shows the response times reported by the DayTrader benchmark application.

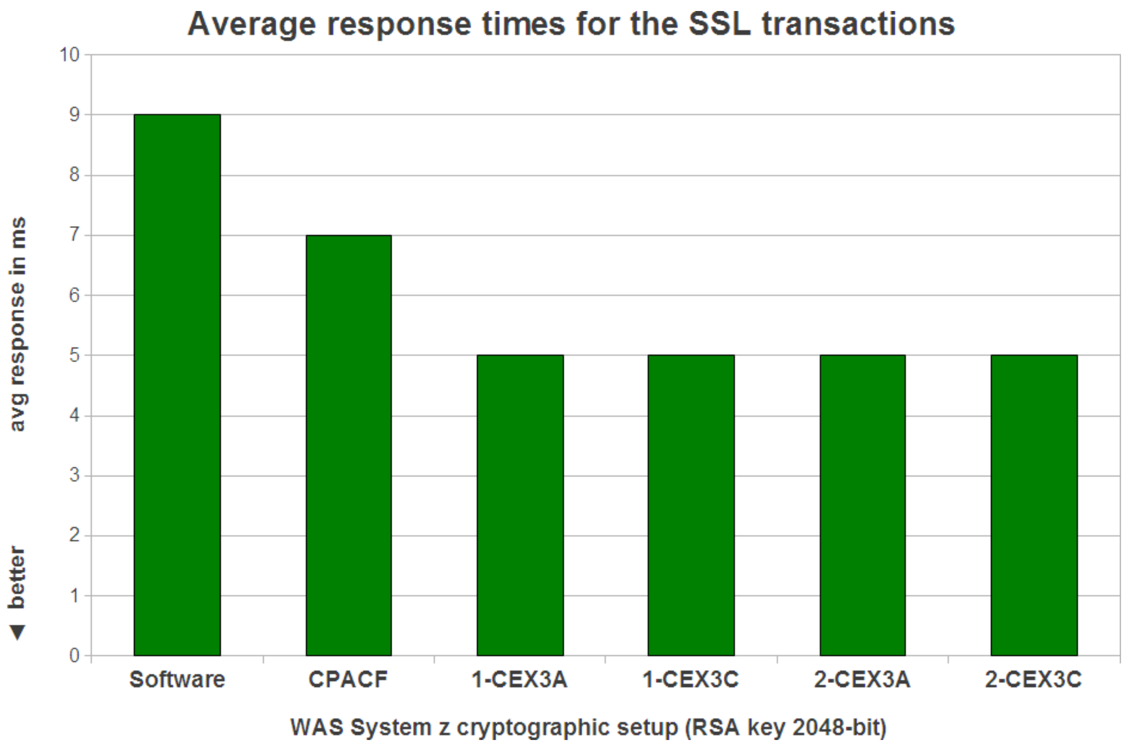


Figure 12. Average response times for the SSL transactions when a 2048-bit RSA key is used

The average response time for a SSL transaction is the amount of time needed to answer a client request. The metric is measured in milliseconds (ms) for the Daytrader benchmark application. Lower numbers mean faster response times.

Observation
The average response times shortens more than 40% when IBM System z cryptographic features are used.

Conclusion
The setups with CEX3 features improves the response times dramatically. All cryptographic operations were processed faster when offloaded to a cryptographic hardware feature. This results in better response times for SSL transactions.

RSA key size 4096-bit

Results for tests performed for scenario 1 using RSA key size 4096-bit are discussed in this topic.

DayTrader transaction throughput and IBM WebSphere Application Server LPAR CPU load

Figure 13 shows the normalized DayTrader SSL transaction throughput when scaling the cryptographic setup and using a 4096-bit RSA key.

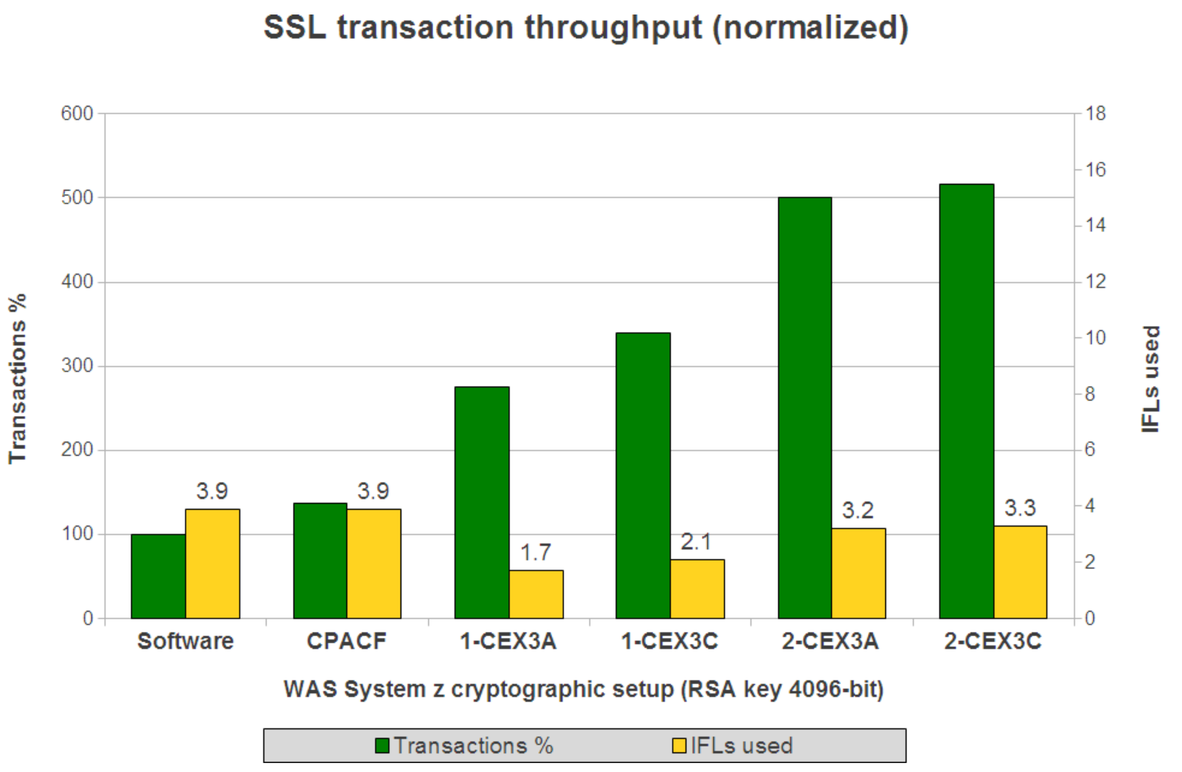


Figure 13. SSL transaction throughput when a 4096-bit RSA key is used

One series of green (dark) bars in Figure 13 show the DayTrader SSL transaction throughput for each considered cryptographic setup. The throughput number for the 'software' setup defines the 100% baseline for the other cryptographic setups.

Observation

The transaction throughput can be nearly tripled with one Crypto Express3 (CEX3) processor. When using two CEX3 processors of the same type, a five times higher throughput rate can be hit.

Conclusion

By using CP Assist for Cryptographic Function (CPACF) the absolute throughput increase is about 30%. When a CEX3 feature is used in the cryptographic setup, the possible transaction rates increase considerably. The selected strong WAS cipher suite together with a RSA key size of 4096-bit require a lot more CPU resources for the cryptographic operations. If the 4096-bit cryptographic operations are calculated completely in software, the available CPU resources are fully used quickly. Hence offloading the cryptographic operations to any System z cryptographic features allows much higher transaction throughput rates.

For the "1-CEX3A/C" setups a high pending request count (5-8 requests) was observed. The maximum pending request queue size for the zcrypt device driver is currently 8. This is an indicator that a single CEX3 processor operates at its limit. Adding a second CEX3 processor (2-CEX3A/C setups) of the same processor type allows a further throughput increase. The zcrypt device driver does a load-balancing over the two available CEX3 processors, which helps to overcome the single CEX3 processor limit in that case.

The second series of bars (yellow/light bars) lists the number of Integrated Facility for Linux (IFL) used on the WAS LPAR, where the LPAR is equipped with 4 IFLs in total.

Observation
All 4 CPUs are fully loaded for the "Software" and "CPACF" setups. Approximately 2 out of 4 IFLs are used for the 1-CEX3 and more than 3 out of 4 IFLs for the 2-CEX3 setups.

Conclusion
For the setups with IBM System z cryptographic features, CPU time is freed due to offloading cryptographic operations to the cryptographic features. The freed CPU time can be used to handle a larger amount of transaction requests compared to the "software" setup.

One CEX3 processor is fully used in the "1-CEX3" setups, tripling the transaction throughput compared to "Software". It is the limiting resource in that case. A second CEX3 processor helps to overcome the single CEX3 processor limit. The fact that the CPU load is below 4 IFLs shows that even in the scenarios with the CEX3 cards, the system is waiting for the cryptographic operations to complete.

DayTrader transaction CPU costs

Another important aspect with respect to the throughput numbers reached is the amount of CPU required to drive the workload, which is considered as cost factor. This metric is calculated as the amount of CPU needed to drive a certain amount of transactions. Figure 14 shows the CPU costs when a 4096-bit RSA key is used.

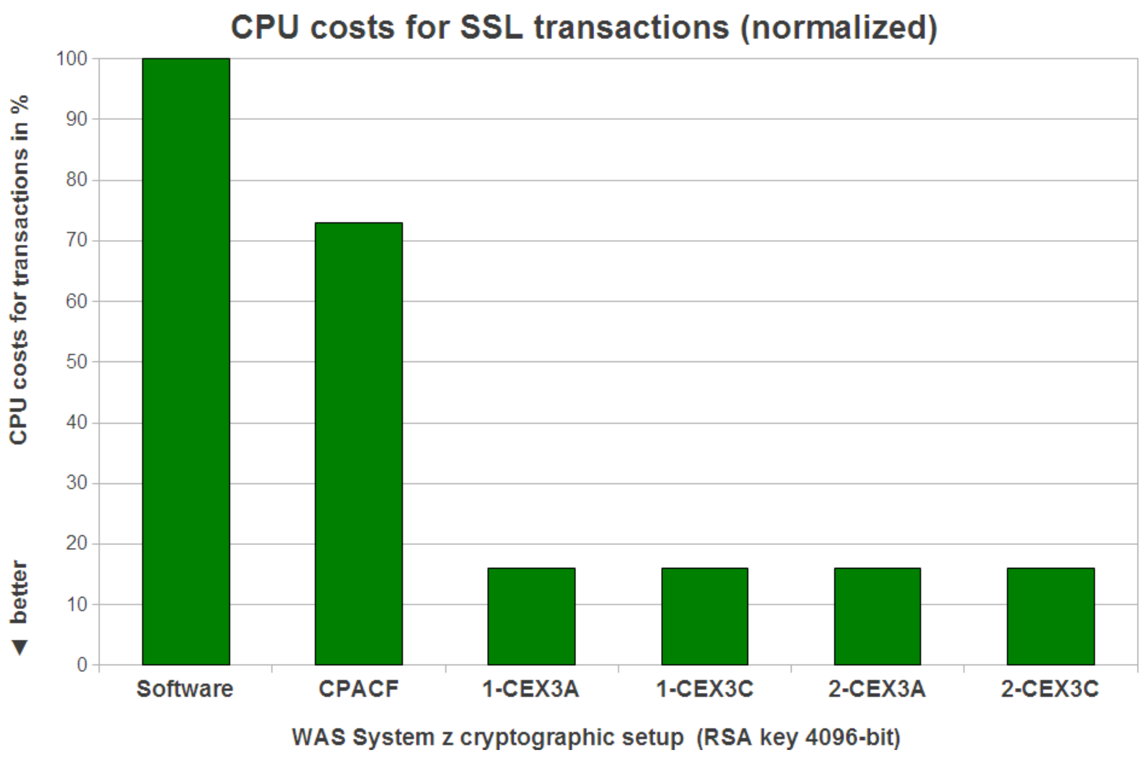


Figure 14. CPU costs for SSL transactions when a 4096-bit RSA key is used

The chart shows the CPU costs for the SSL transactions when using a 4096-bit RSA key. The numbers are normalized against the 'Software' CPU costs number.

Observation
The use of IBM System z cryptographic features reduces the CPU costs for SSL transactions. For the setups including CEX3 features, the CPU costs for transactions are reduced by more than factor six.

Conclusion
The exploitation of the IBM System z cryptographic features reduce the CPU costs for SSL transactions dramatically. The reduced CPU costs allow more incoming transactions to be processed and a higher transaction throughput rate is possible.

DayTrader average response times

Figure 15 shows the response times reported by the DayTrader benchmark application.

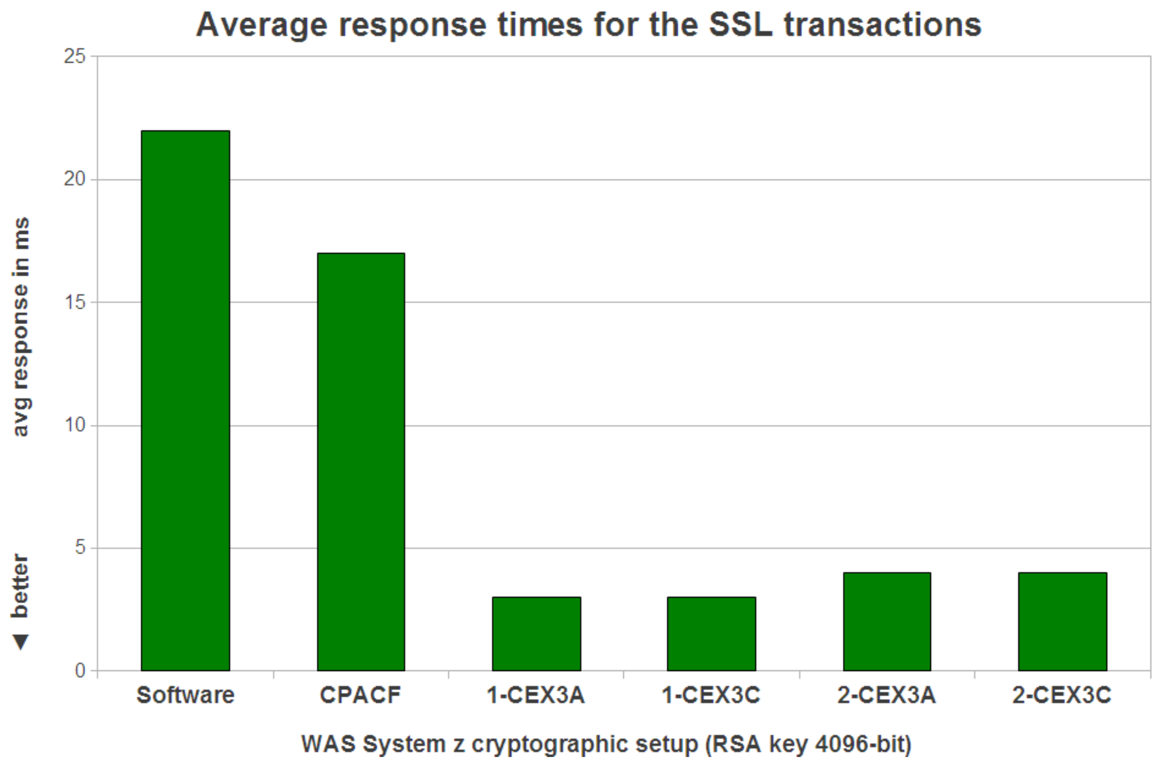


Figure 15. Average response times for the SSL transactions when a 4096-bit RSA key is used

The average response time for a SSL transaction is the amount of time needed to answer a client request. The metric is measured in milliseconds (ms) for the DayTrader benchmark application. Lower numbers mean faster response times.

Observation
The average response time is below 5 ms for the measurement series with CEX3 features. Whereas the average response time for the 'Software' setup is over 20 ms.

Conclusion
The cryptographic setups with CPACF and CEX3 features dramatically improves the response times compared to software encryption. All cryptographic operation were processed much faster when offloaded to a cryptographic hardware feature. This results in considerably better response times for the applications running on a WAS server. The cryptographic hardware on Linux on System z allows to provide a very high level on security at a low impact on performance.

Results for scenario 2: IBM HTTP server

For scenario 2 the IBM HTTP server (IHS) web server drives the SSL connection.

This means that the HTTPS requests are processed by the IHS web server and IBM WebSphere Application Server (WAS) executes the application logic. IHS runs on the same Linux server as WAS and the communication between IHS and WAS is not encrypted. The self-signed certificate was stored in the PKCS#11 cryptographic ICA token and was defined with a key size of 2048-bit, where the scaling is done over the cryptographic setups. A key size of 4096-bit was chosen in a second measurement series.

Note: The workload generated (number of virtual users) by the workload driver is kept constant across all measured scenarios.

See System under test (SUT) overview for a detailed description of the System Under Test (SUT) scenarios

RSA key size 2048-bit

Results for tests performed for scenario 2 using RSA key size 2048-bit are discussed in this topic.

DayTrader transaction throughput and IBM WebSphere Application Server LPAR CPU load

Figure 16 shows the normalized DayTrader SSL transaction throughput, when scaling the cryptographic setup and using a 2048-bit RSA key.

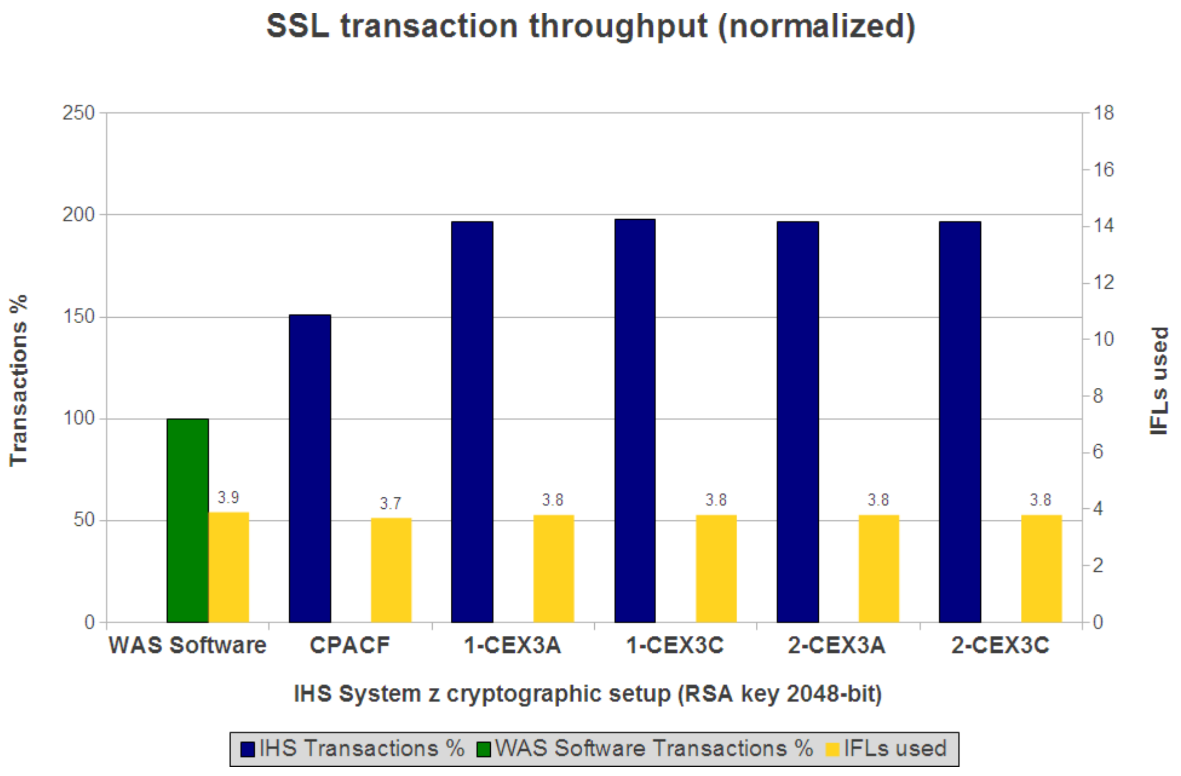


Figure 16. SSL transaction throughput when a 2048-bit RSA key is used

The green bar and the blue bars (dark) in the diagram show the DayTrader SSL transaction throughput for each considered cryptographic setup. The throughput number from the "IBM WebSphere Application Server (WAS) Software" setup (green bar) defines the 100% baseline for the other cryptographic setups.

Observation
The transaction throughput can be doubled when using the IBM System z cryptographic features.

Conclusion
Approximately 50% of the throughput increase can be attributed to CP Assist for Cryptographic Function (CPACF) and another 50% to Crypto Express3 (CEX3) for this type of workload. A second CEX3 processor does not result in higher throughput numbers, because a single CEX3 processor is sufficient to handle the workload. For the 2-CEX3 setups the zcrypt device driver does a load-balancing among the available CEX3 processors.

The WAS LPAR where the IBM HTTP server (IHS) runs is equipped with 4 Integrated Facilities for Linux (IFLs).

Observation
All 4 IFLs are almost fully used on the WAS LPAR across all cryptographic setups in this scenario (see yellow/light bars). Additional application benchmark runs with a reduced number of clients and hence less CPU load showed similar ratios for the scaling factor of the cryptographic setups.

Conclusion
For the setups with IBM System z cryptographic features, CPU time is freed due to offloading cryptographic operations to the cryptographic features. The freed CPU time can be used to handle a greater amount of transaction requests compared to 'WAS Software' setup. This leads to a higher SSL transaction throughput number in total for the setups exploiting the IBM System z cryptographic features.

DayTrader transaction CPU costs

Another important aspect with respect to the throughput numbers reached is the amount of CPU required to drive the workload, which is considered as a cost factor. This metric is calculated as the amount of CPU needed to drive a certain amount of transactions. Figure 17 shows the CPU costs, when a 2048-bit RSA key is used.

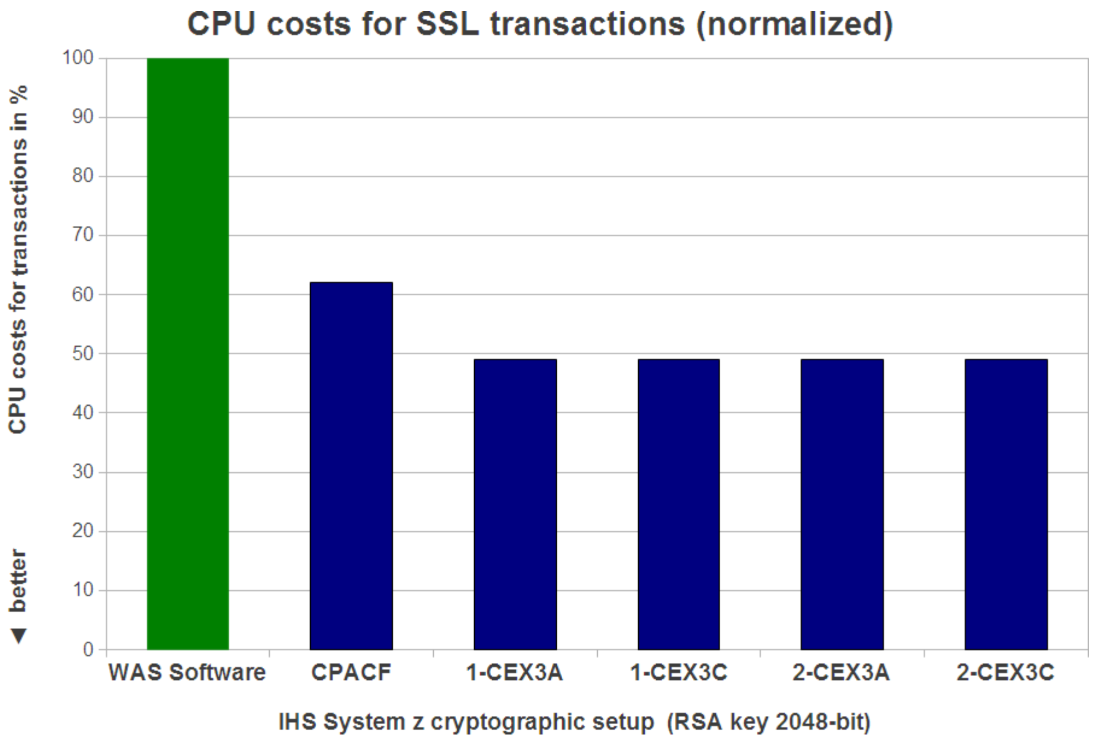


Figure 17. CPU costs for SSL transactions when a 2048-bit RSA key is used

The chart shows the CPU costs for the SSL transactions when a 2048-bit RSA key is used. The numbers are normalized against the 'WAS Software' CPU costs number.

Observation
The CPU costs for the transactions are nearly half as much for the CEX3 measurement series.

Conclusion
The exploitation of the IBM System z cryptographic features reduces the CPU costs for SSL transactions dramatically. The reduced CPU costs allow more incoming transactions to be processed and a higher transaction throughput rate is possible.

DayTrader average response times

Figure 18 shows the response times reported by the DayTrader benchmark application.

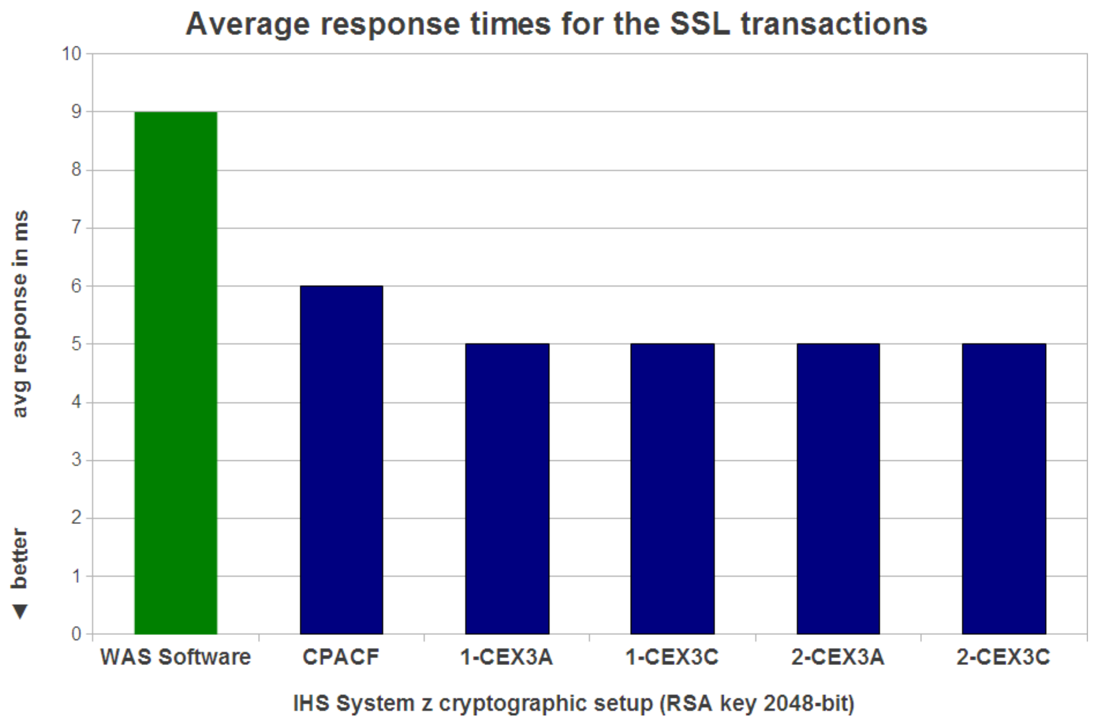


Figure 18. Average response times for the SSL transactions when a 2048-bit RSA key is used

The average response time for a SSL transaction is the amount of time needed to answer a client request. The metric is measured in milliseconds (ms) for the DayTrader benchmark application. Lower numbers mean faster response times.

Observation
The average response times shortens more than 40% when IBM System z cryptographic features are used.

Conclusion
The setups with CEX3 features dramatically improve the response times compared to software encryption. The cryptographic operations can be processed much faster when offloaded to a cryptographic hardware feature. This results in better response times for SSL transactions.

RSA key size 4096-bit

Results for tests performed for scenario 2 using RSA key size 4096-bit are discussed in this topic.

DayTrader transaction throughput and IBM WebSphere Application Server LPAR CPU load

Figure 19 shows the normalized DayTrader SSL transaction throughput, when scaling the cryptographic setup and using a 4096-bit RSA key.

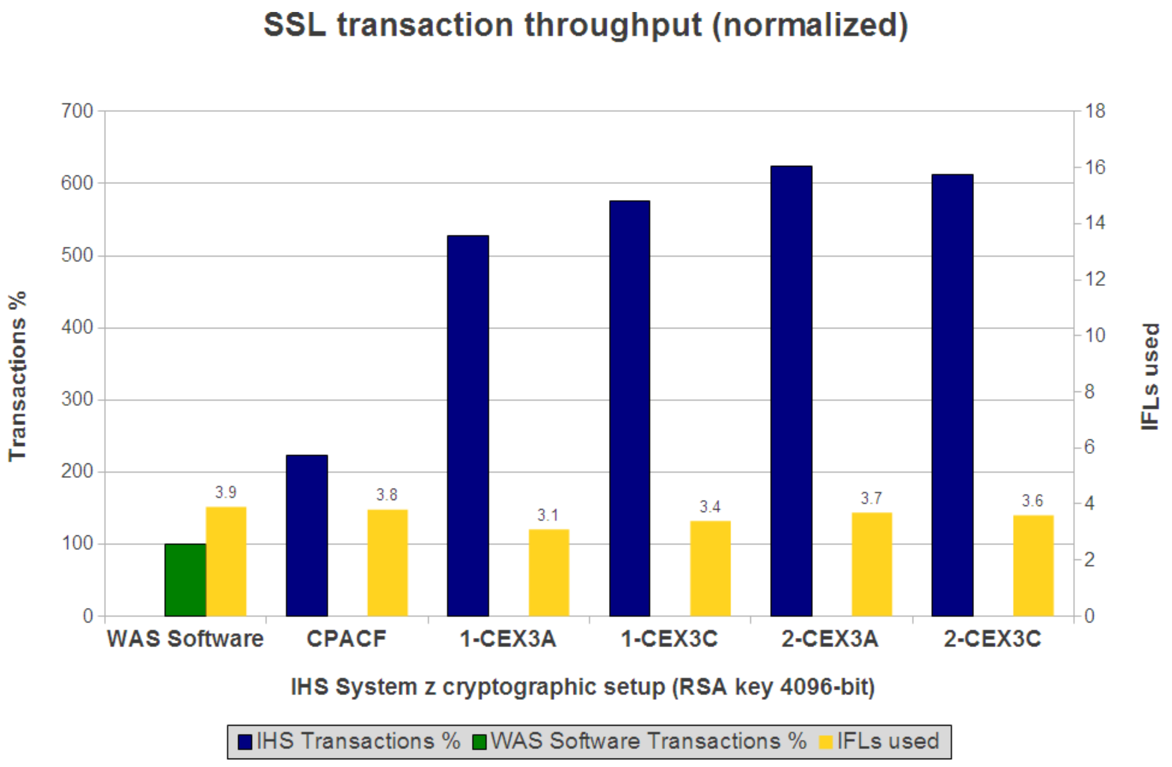


Figure 19. SSL transaction throughput when a 4096-bit RSA key is used

The green bar and the blue bars (dark) in the diagram show the normalized DayTrader SSL transaction throughput for each considered cryptographic setup. The throughput number for the "IBM WebSphere Application Server (WAS) software" setup (green bar) defines the 100% baseline for the other cryptographic setups.

Observation
The use of CP Assist for Cryptographic Function (CPACF) doubles the transaction throughput compared to a pure software setup. The transaction throughput can be increased by more than five times with one Crypto Express3 (CEX3) processors. When two CEX3 processors of the same type are used, a six times higher throughput rate can be achieved.

Conclusion
When a CEX3 feature is used in the cryptographic setup, the possible transaction rates increase considerably. The selected strong WAS cipher suite together with a RSA key size of 4096-bit require a lot more CPU resources for the cryptographic operations. If the 4096-bit cryptographic operations are calculated completely in software, the available CPU resources are fully used quickly. Hence offloading the cryptographic operations to any IBM System z cryptographic features allows much higher transaction throughput rates.

For the '1-CEX3A/C' setups a high pending request count (5-8 requests) can be observed. The maximum pending request queue size for the zcrypt device driver is currently 8. This is an indicator that a single CEX3 processor operates at its limit. Adding a second CEX3 processor (2-CEX3A/C setups) of the same processor type allows a further throughput increase. The zcrypt device driver does a load-balancing over the two available CEX3 processors, which helps to overcome the single CEX3 processor limit in that case.

The second series of bars (yellow/light bars) lists the number of Integrated Facility for Linux (IFLs) used on the WAS LPAR, where the LPAR is equipped with 4 IFLs in total.

Observation
All 4 CPUs are almost fully loaded for the "WAS Software" and "CPACF" setups. Approximately 3 out of 4 IFLs are used for the 1-CEX3 and close to 4 IFLs for the 2-CEX3 setups.

Conclusion
For the setups with IBM System z cryptographic features, CPU time is freed because cryptographic operations are offloaded to the cryptographic features. The freed CPU time can be used to handle a larger amount of transaction requests compared to the "Software" setup.

One CEX3 processor is fully used in the '1-CEX3' setups, almost sextupling the transaction throughput compared to "Software"'. It is the limiting resource in that case.

A second CEX3 processor helps to overcome the single CEX3 processor limit and nearly all 4 IFLs are now used.

Daytrader transaction CPU costs

Another important aspect with respect to the throughput numbers reached is the amount of CPU required to satisfy the workload, which is considered as cost factor. This metric is calculated as the amount of CPU needed to drive a certain amount of transactions. Figure 20 shows the CPU costs when a 4096-bit RSA key is used.

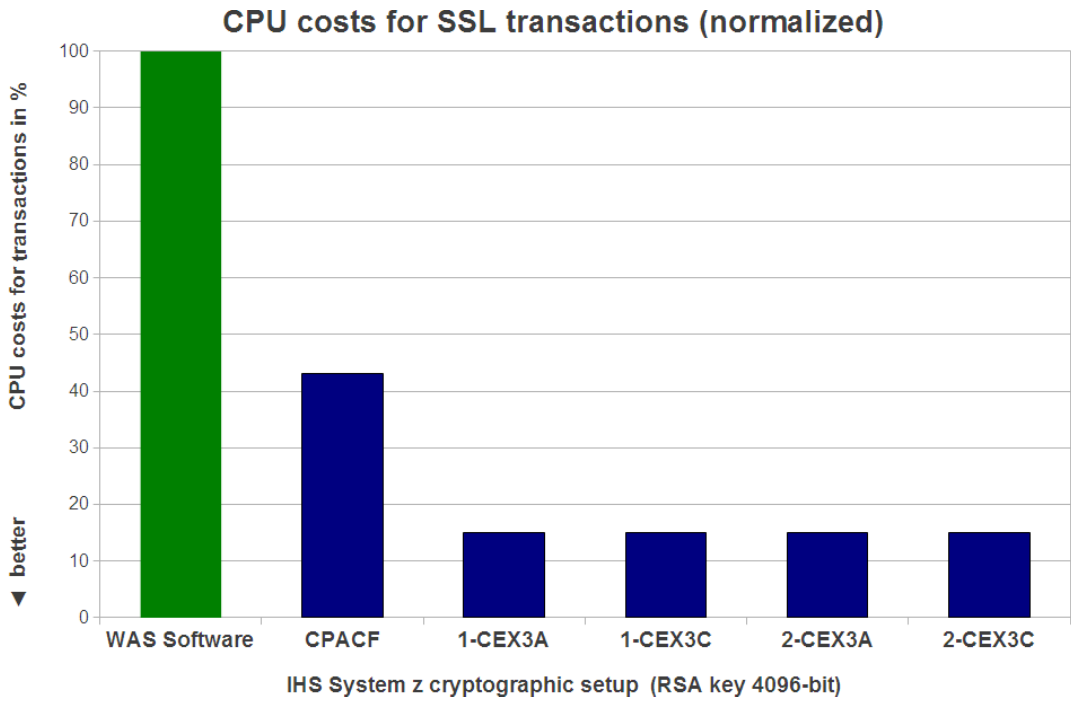


Figure 20. CPU costs for SSL transactions when a 4096-bit RSA key is used

The chart shows the CPU costs for the SSL transactions when using a 4096-bit RSA key. The numbers are normalized against the "WAS Software"' CPU costs number.

Observation
The usage of System z cryptographic features is reducing the CPU costs for SSL transactions. For the setups including CEX3 features, the CPU costs for transactions are reduced by more than factor six.

Conclusion
The exploitation of the System z cryptographic features reduce the CPU costs for SSL transactions dramatically. The reduced CPU costs allow to process more incoming transactions and hence a higher transaction throughput rate is possible.

DayTrader average response times

Figure 21 shows the response times reported by the DayTrader benchmark application.

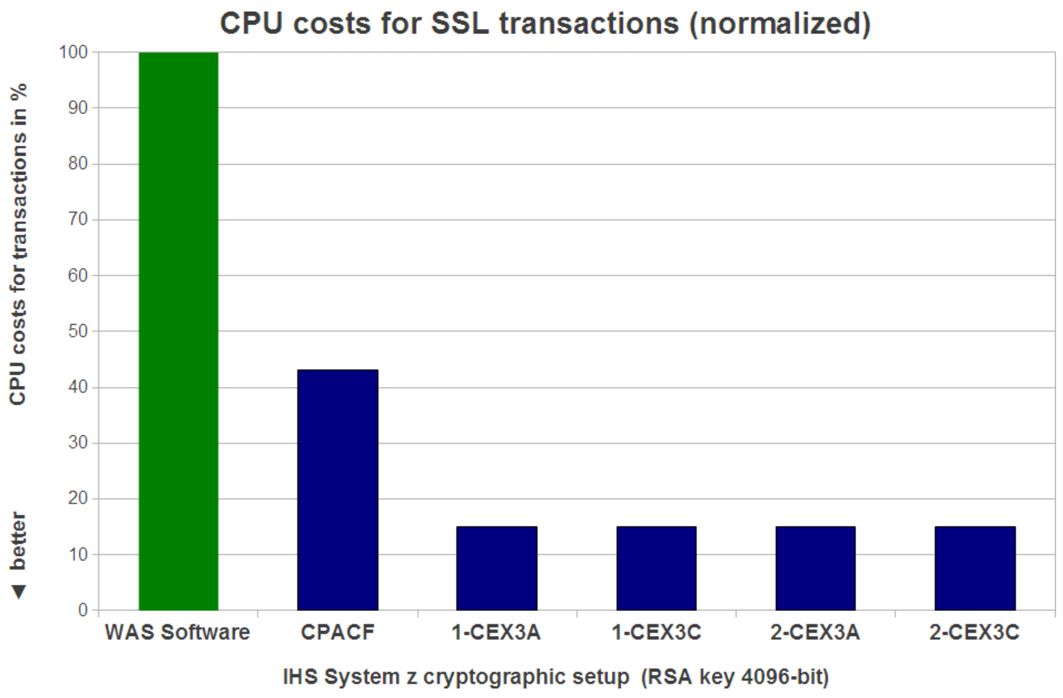


Figure 21. Average response times for the SSL transactions when a 4096-bit RSA key is used

The average response time for a SSL transaction is the amount of time needed to answer a client request. The metric is measured in milliseconds (ms) for the Daytrader benchmark application. Lower numbers mean faster response times.

Observation
The average response time is about 5 ms for the measurement series with CEX3 features. Whereas the average response time for the "WAS Software" setup is over 20 ms.

Conclusion
The cryptographic setups with CPACF and CEX3 features provided the best response times in our setup. Any cryptographic operation can processed much faster when offloaded to a cryptographic hardware feature. This results in considerably better response times for the applications running on a WAS server.

References

IBM documentation and articles:

- David Hare, Staff Software Engineer, IBM
  Christoper Blythe, Advisory Software Engineer, IBM, (Update June 22, 2011)
  Case study: Tuning WebSphere Application Server V7 and V8 for performance
  http://www.**ibm.com**/developerworks/websphere/techjournal/0909_blythe/0909_blythe.html

- Ming Xia, Software Engineer, IBM, (Nov 10, 2009)
  A performance benchmark method for comparing open source Java application servers
  http://www.**ibm.com**/developerworks/opensource/library/os-perfbenchmk/index.html

- IBM support portal, (May 19, 2010)
  Enabling and configuring cryptographic technology with WebSphere Application Server V7.0 on Linux for System z hardware
  http://www.**ibm.com**/support/docview.wss?uid=swg27017055

- M. A. Tebolt , IBM Platform Evaluation Test Laboratory Poughkeepsie (Dec, 2009)
  Configuring WebSphere V7.0 and IBM HTTP Server V7.0 to use Cryptographic Hardware for SSL Acceleration on Linux on IBM System z
- IBM Linux for System z HiperSockets documentation
  HiperSockets Implementation Guide, SG24-6816
- IBM Linux for System z PAV and HyperPAV documentation
  How to Improve Performance with PAV, SC33-8414
- IBM Linux for System z Device Drivers documentation
  Linux on System z - Device Drivers, Features, and Commands SC33-8411-13
  http://www.**ibm.com**/support/docview.wss?uid=pub1sc33841113

- Details about Linux for System z network tuning
  http://www.**ibm.com**/developerworks/linux/linux390/perf/tuning_networking.html

- System z Input/Output Configuration user's guide
  Input/Output Configuration Program User's Guide for your mainframe system, SB10-7037-10
- IBM WebSphere Application Server Information Center (online)
  http://pic.dhe.**ibm.com**/infocenter/wasinfo/v8r0

- IBM DB2 Database for Linux, UNIX, and Windows Information Center (online)
  http://pic.dhe.**ibm.com**/infocenter/db2luw/v9r7

Other documentation and download sites:

- Apache Geronimo v3.0 - DayTrader - a more complex application
  http://cwiki.apache.org/GMOxDOC30/daytrader-a-more-complex-application.html

- IBM developerWorks® Java security information site:
  - Java unlimited policy files download site
  - IBMPKCS11Impl Provider Guide download site
    http://www.**ibm.com**/developerworks/java/jdk/security/

Notices

April 2013

ZSW03250-USEN-00