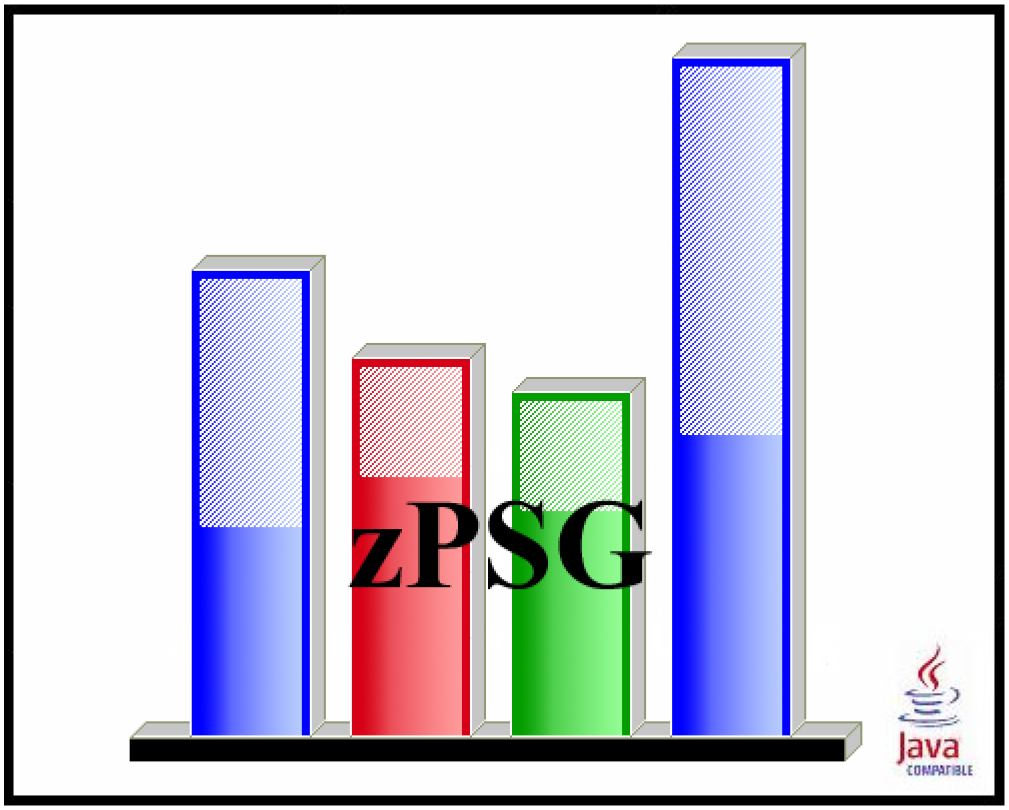
Processor Selection Guide
for
IBM System z

zPSG User's Guide for WebSphere Application Server



Version 5.9a
zPSG WAS UG V59a 2014a01.doc
February 26, 2014

WebSphere Application Server

z/OS and Linux

This tool provides estimates of System z processor capacity for J2EE EJB 3.0 64-bit applications running under WebSphere Application Server (WAS) 8.0 for z/OS and Linux using JCA (J2C) connectors. J2EE and JCA/J2C are standards for Java application design and deployment.

See the WAS Glossary of Terms for a definition of terms used in describing WAS and Java web applications.

The general approach to sizing WAS applications on System z is to describe the content of a typical transaction and provide a transaction rate for an interval of peak demand. The usual span of time for a peak interval is 15 minutes, and you want to specify the average transactions/second for that interval. Note that if you have statistics for the average transaction rate for prime shift or for a day or week, you might want to apply a peak-to-average multiplier factor to averages for long periods of time to arrive at an average rate for a 15 minute interval.

To describe the content of a typical transaction, a list of potential functions or activities that can comprise a WAS transaction are presented on an input screen. For a rough sizing estimate, you may simply check the activities to be included in the typical transaction and accept all of the defaults related to each activity. The only functions that don't have defaults are the characterization of Java Application Processing, DB2 SQL processing, and DB2 access (activities #6, 8, and 9). These activities represent the customer's business transaction processing so there are no defaults and input is required. Refer to the *WAS Application Definition* window to see the list of activities that can be included in a typical transaction.

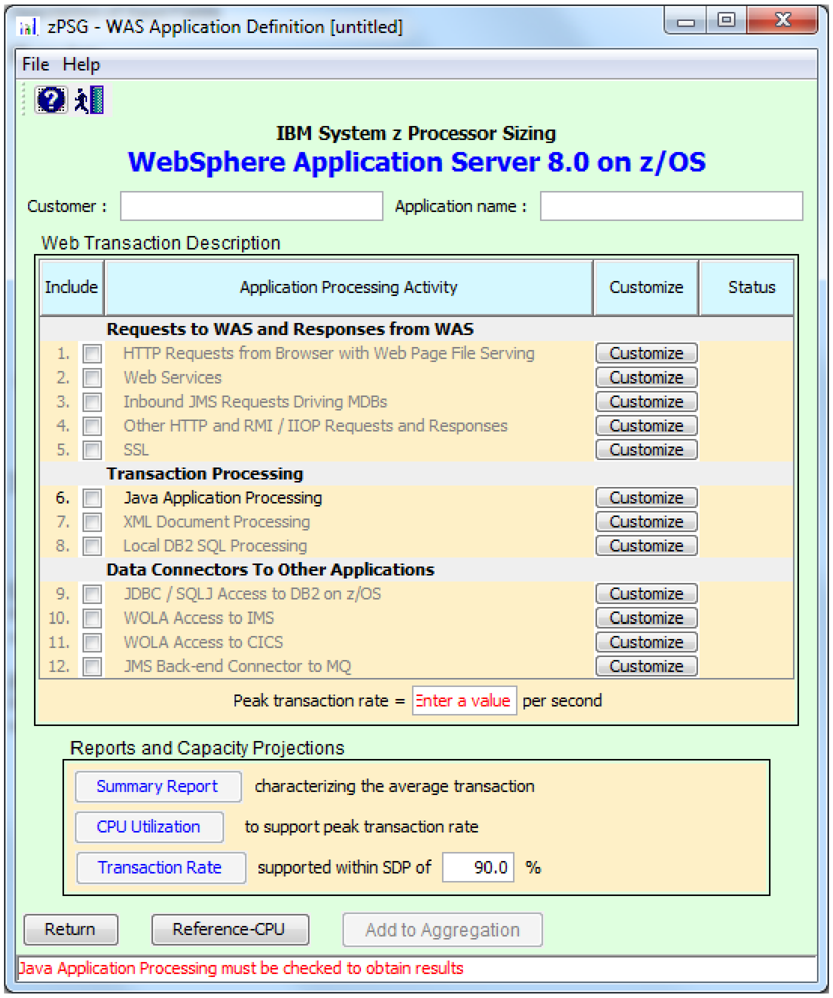
How To Do a WAS Sizing

When you select WAS sizing support from the ***Product Selection*** window to begin a new sizing, the ***WAS Application Definition*** input screen is presented. When you first access the primary screen, the only activity that is enabled for selection is Java Application Processing (activity #6). A message in the status bar at the bottom of the window explains that Java processing must be selected. Since all WAS J2EE applications are based on Java processing you must include this activity in all sizings. When Java Application Processing is included, a window pops up to enable you to rate the customer's application processing on a scale of 1 to 10. Select a Java application complexity rating for the customer's typical transaction by moving the slider bar across the scale of 1 to 10. When you are finished, click on the **Return** button to go to the ***WAS Application Definition*** window to specify other activities or functions to be included in a transaction. Check the **Include** boxes for all activities that are part of a typical WAS transaction. Do not include activities that occur only occasionally or are not part of a typical transaction running during an interval of peak demand.

If you want to over-ride details about any of the activities that you have included, click on the **Customize** buttons to access windows for in-depth input on each activity. Remember that even if you click on a **Customize** button and change something in the subsequent input screen, this input will not be reflected in the sizing estimate unless you include that activity on the ***WAS Application Definition*** screen. You must include at least 1 of the first 4 activities, since these are the options that reflect transaction requests coming into WAS.

To complete input to a WAS sizing, specify a peak transaction rate and then click on either the **Summary Report** , **CPU Utilization** or **Transaction Rate** buttons in the Reports and Capacity Projections section of the window to see results.

WAS Application Definition


zPSG - WAS Application Definition [untitled]

File Help

IBM System z Processor Sizing

WebSphere Application Server 8.0 on z/OS

Customer : Application name :

Web Transaction Description

Include	Application Processing Activity	Customize	Status
	Requests to WAS and Responses from WAS		
1. ☐	HTTP Requests from Browser with Web Page File Serving	Customize	
2. ☐	Web Services	Customize	
3. ☐	Inbound JMS Requests Driving MDBs	Customize	
4. ☐	Other HTTP and RMI / IIOP Requests and Responses	Customize	
5. ☐	SSL	Customize	
	Transaction Processing		
6. ☐	Java Application Processing	Customize	
7. ☐	XML Document Processing	Customize	
8. ☐	Local DB2 SQL Processing	Customize	
	Data Connectors To Other Applications		
9. ☐	JDBC / SQLJ Access to DB2 on z/OS	Customize	
10. ☐	WOLA Access to IMS	Customize	
11. ☐	WOLA Access to CICS	Customize	
12. ☐	JMS Back-end Connector to MQ	Customize	

Peak transaction rate = Enter a value per second

Reports and Capacity Projections

Summary Report characterizing the average transaction

CPU Utilization to support peak transaction rate

Transaction Rate supported within SDP of 90.0 %

Return Reference-CPU Add to Aggregation

Java Application Processing must be checked to obtain results

This window is displayed when the **Size Workload** button is clicked on the ***Product Selection*** window when **WebSphere Application Server** has been selected for z/OS or Linux (the example used here is for z/OS).

Note: Java Application Processing must be included in a sizing estimate. And you must include 1 of the first 4 activities under Requests to WAS and Responses from WAS, and enter a value in the Peak transaction rate = ' ' per second field to get a sizing estimate.

Description of Input Fields

Menu bar

File

New	Start a new study. Sets all fields to initialization values.
Load	Load a previously saved study **Note: A study file saved from a version of zPSG prior to 5.8 is incompatible and it can't be loaded.**
Save	Save the current study
Save as	Save the current study as a new file
Exit	Exit window and return to the ***Product Selection*** window (Ctrl-E)
Exit zPSG	Terminate **zPSG** execution (Ctrl-Q). Exit zPSG can also be invoked from the Exit zPSG button on the tool bar.

Help

Context Help (F1)	Help for this window
About zPSG	Product information

Toolbar

? button	Click this button to go to Help for this window.
Exit zPSG button	Click this button to terminate **zPSG** execution.

Customer =
Input field, for documentation purposes, not required. If you want to save a copy of the sizing estimate, you can use this field to document which sizing it is.

Application name =
Input field, for documentation purposes. This field is only required when the workload will be added to the aggregation candidate list using the **Add to Aggregation** button.

Web Transaction Description

Capacity requirements are estimated from a description of the activity involved in a **typical WAS (or web) transaction**, multiplied by a transaction rate. The scope of a web transaction is flexible and up to you. Nothing inherent in WAS prescribes the definition or limits of a transaction. It can involve multiple user interactions with WAS, multiple web pages, multiple servlets and EJBs, and multiple data connectors. The main guideline is that a web transaction should represent a repeatable set of activity. Frequently, a business operation is used as a web transaction. If you want to define a web transaction that involves multiple WAS requests, servlets, EJBs, and data connectors, just be sure that the transaction rate provided reflects the rate at which the **entire** web transaction occurs, and not "website hits/sec" or CICS/IMS/DB2 trans/sec, etc.

When providing input on your customer's WAS transaction, **do not** include information on any web page files, inbound/outbound data, data connectors, etc. that occur only occasionally and are not part of a typical web transaction. And **do not** include web transactions that don't run during peak demand. The sizing should be based on the work running during a 15 minute interval of overall peak demand for your WAS application. This is what will determine the capacity you need to configure. Any work that runs during other time periods is not relevant.

***Include* column**

Input fields (checkboxes). Since Java application processing is necessary for there to be a WAS J2EE application, the include checkbox for Java Application Processing must be checked when starting a sizing. See instructions, How To Do a WAS Sizing. At least 1 of the first 4 options under Requests to WAS and Responses from WAS must also be included. Click the boxes for each kind of activity in a typical transaction. CPU costs associated with these, **and only these**, items will be included in the estimate. Unless you use the **Customize** buttons to provide detailed input, default values are used for all activities except Java application processing, DB2 SQL processing, and DB2 access, which don't have defaults. These three items reflect your customer's application processing. You must use the **Customize** buttons for Java Application Processing and include 1 of the Request/Response items to get an estimate. If Local DB2 SQL Processing or JDBC / SQLJ Access to DB2 are included in the estimate, you must click the **Customize** buttons to describe the DB2 SQL processing and the DB2 access. The other items are associated with input and output to the web application and to data connectors to other back-end applications (CICS, IMS) or middleware (MQ).

***Application Processing Activity* column**

Not input fields. This is a list of potential types of activity that might be included in a typical web transaction. Click the boxes in the ***Include*** column to include all the items in this list that are part of a typical web transaction that runs during an interval of peak demand. CPU costs for these items, **and only these items**, will be included in the sizing estimate. For more information on each activity, see the following discussion following the description of the ***Customize*** and ***Status*** columns.

Customize column

Input fields. **Customize** buttons are provided to access windows for over-riding defaults for each activity or function.

***Status* column**

Not input fields. Reports the status of each item in the list, whether defaults are being used for a sizing estimate or more detailed input from other windows as the result of using the **Customize** buttons.

Following is a description of each activity or function in the ***Application Processing Activity*** column.

Requests to WAS and Responses from WAS

Description of Activities

Sizing support for WAS includes the following vehicles for inbound requests to WAS and outbound responses from a WAS transaction. These requests/responses are covered by activities #1-4.

HTTP Requests from Browser with Web Page File Serving

This activity covers sizing the CPU requirement for processing requests coming in from a browser via HTTP and for web page file serving as the response. It includes the costs to serve both static files and dynamically-created files using the WAS HTTP Transport Handler. Note that not all static files used to compose a web page are served from WAS; see Considerations section below or discussion for the in-depth window on web page file serving for more insight on this subject. CPU costs for serving static files include the cost to retrieve the files from disk and network transmission costs. CPU costs for dynamic files include the cost of transmitting any inbound data (which usually accompany requests for dynamic files) and the JSP cost to create (format) the file as well as the transmission cost to serve the file.

Note: For static file serving on z/OS, the sizing assumes use of FRCA.

Web Services

See the Help section on the Web Services window for an explanation of what web services messages are. Sizing support includes the cost of transmitting and processing both inbound web services request messages and/or outbound response messages, based on the JAX-WS web services specification available in current releases of WAS.

This activity does not include XML documents that are **not** part of a web services message. See Other HTTP and RMI/IIOP Requests and Responses below for input on non-web services XML document transmission costs.

Inbound JMS Requests Driving MDBs

These requests involve the CPU cost of receiving an inbound JMS message into WAS and executing a Message Driven Bean (MDB), which typically initiates a WAS application transaction. Note that since JMS messaging implies an asymmetric programming model, it is not assumed that that an input message is paired with an outgoing response of some kind. No CPU is included for a response in the cost of Inbound JMS Requests Driving MDBs. You must define any response from the WAS transaction separately.

Other HTTP and RMI / IIOP Requests and Responses

These requests are usually program-to-program requests, coming in via either HTTP or RMI / IIOP from a browser, client Java application or another WAS, possibly with an outbound response. Often in request/response pairs, they may involve transmission and processing of accompanying inbound and outbound data. These requests can be used to account for communication between WebSphere Portal Server portlets and a WAS application running behind the portal that you want to size (WPS sizing support, available from Techline, includes some general consideration of portlet processing, but does not include sizing significant applications behind the portal. Those applications must be sized separately.)

In addition to the above 4 request/response options, you can specify whether or not transaction input and output are SSL protected in activity #5.

SSL

Sizing SSL capacity requirements on System z general purpose CPs or IFLs is supported. Sizing support for crypto assist hardware is assumed. We provide estimates based on the use of JSSE for z/OS and Linux. Support for the TDES/SHA and AES/SHA ciphers/hashing algorithms is available. Use of SSL can contribute significantly to the capacity requirement for a WAS application.

You may include any or all of the first 4 Request/Response options, although it would be somewhat unusual to have web services and the other options in the same transaction. You must include at least 1 of the first 4 Request/Response options.

Defaults

Defaults for **HTTP Requests from Browser with Web Page File Serving** are:

  - 1 HTTP request per web page file served. Each request for dynamic files is accompanied by 256 bytes of inbound data.
  - Light web page file serving activity: 1 static file of 10k and 1 dynamic file of 4k with Simple JSP processing served per web page
  - 2 page pages served per transaction
  - If SSL is included, all inbound data and web page files are SSL protected.

- Defaults for **Web Services** are:
  - 1 inbound request message plus 1 outbound response message.
  - Message size is 2k, message complexity is medium (we do not currently have performance data to support sizing web services messages of different levels of complexity)
  - If SSL is included, all messages are protected.
- Defaults for **Inbound JMS Requests** are:
  - Point-to-point messaging in TCP/IP mode
  - Use of WebSphere MQ product (not the WAS internal java messaging function)
  - 1 inbound message of 2k driving an MDB with Simple complexity
  - Message is not automatically paired with an outbound response – any response must be specified separately
  - If SSL is included, all messages are protected.
- Defaults for **Other HTTP and RMI/IIOP Requests and Responses** are:
  - 1 HTTP request/response pair per tran
  - Each request is accompanied by 0.5k of inbound data and each response involves sending 1k of outbound data to requestor
  - If SSL is included, all inbound and outbound data is protected.

- Defaults for **SSL** are:
  - Use of crypto assist hardware for full (non-cached) SSL handshakes. (Cached handshakes do not require hardware assist.)
  - Average number of transactions per session is 5000
  - Keep-alive connection is being used
  - If SSL is selected, TDES/SHA cipher and hashing algorithm
  - SSL protection for all inbound/outbound data, messages, files, etc. included in the transaction.

Considerations

- Our sizing support for receiving requests and serving web page files is currently based on interfacing with WAS via the WAS HTTP Transport Handler, not the separate IBM HTTP Server (IHS). While we don't have specific performance data on the IHS plug-in to WAS, we know that the pathlength is longer than the WAS Transport Handler for receiving requests into WAS and serving dynamic files, and shorter for serving static files.
- The number of static web page files served from WAS is generally less than the number of files used by the browser to compose the web page. Many small, frequently used static files are cached by the browser and not served as part of a web transaction; other static files are cached on an Edge Server and not served from System z. For this reason, we are assuming light file serving activity by default, but providing options for heavier file serving via the **Customize** button.
- Sizing support for web services assumes the significantly-improved performance of web services, first available in WAS 5.0.2.
- XML document processing capacity requirements are covered by activity #7, XML Document Processing, under the Transaction Processing section because, although XML documents can be used as part of requests to WAS and responses from WAS, they can also be used for inter-application communication in the same system image and not transmitted over the network. For this reason, we separate XML processing (parsing, XSL transformation, etc.) from the CPU costs associated with transmitting XML documents over the network. If you have XML documents that are transmitted over the network, you have to describe the size of inbound and outbound documents in activity #4, Other HTTP and RMI/IIOP Requests and Responses.
- **zPSG** supports 2 of the several SSL ciphers/hashing algorithms available, mostly because the others are not widely used. The most popular cipher is Triple DES/SHA (TDES). TDES, used by banks, financial institutions and government agencies requiring very high levels of encryption security, is supported by crypto assist hardware, for both the full SSL handshake and encryption/decryption of data. TDES/SHA is the default for SSL. The AES/SHA cipher/hashing algorithm is also supported. You would not want to use TDES or AES without crypto hardware.

Transaction Processing

Description of Activities

Sizing support for transaction processing includes 3 activities, which constitute the customer's business logic in a transaction: Java application processing, XML document processing and access to local DB2 plus SQL statement processing.

When Java Application Processing, Local DB2 SQL Processing or JDBC / SQLJ Access to DB2 are first included, the windows for configuring them will be automatically displayed. Once you have rated the Java and DB2 processing, if you want to review or change the ratings, click the **Customize** buttons. For more information on selecting the ratings, see the help sections for the Java Application Processing, Local DB2 SQL Processing or the JDBC / SQLJ Access to DB2 windows.

Java Application Processing

This activity must always be included in a WAS sizing since you cannot have a J2EE WAS transaction without Java processing. When you select this activity, a window is displayed to rate your customer's Java processing on a linear scale of 1 to 10. Move the slider to select a rating. Profiles of two common workloads used in WAS lab measurements are shown to help you select a complexity rating. See Java Processing Workload Profiles for a description of the workloads. There is no default rating.

XML Document Processing

XML documents are generally used for inter-application communication. They can be received by WAS with HTTP requests, sent by WAS to a client, or not transmitted over the network at all but received from or passed to another application running in the same system image. If XML documents are transmitted over the network, you will need to include the size of the documents as inbound and/or outbound data under activity #4, Other HTTP and RMI/IIOP Requests and Responses to cover the CPU cost of receiving and converting the data to run in a WAS Java environment or to send out an outbound document. You may also have an application in which XML processing occurs for documents that are not sent over the network. Inbound documents to WAS may or may not involve XML parsing and parsing may or may not involve validation. Outbound documents may be created by JSPs or by XSL Transformation, or they may be created from a DOM tree. They could also be retrieved from DB2 or received back from CICS or IMS.

XML processing varies greatly depending on factors like the size and complexity of the document, whether parsing and/or validation are included, etc. The number of attributes per k of data is an important factor in determining the complexity of an XML document when using the SAX parser. Although using a Java program to create an XML document does not really involved XML processing, it is included in this section because an XML document is the result of this processing. For details on characterizing XML processing, see the Help section for XML Document Processing.

Note: Web services messages include XML documents but are described in the Web Services section. Do not describe web services messages as XML documents for sizing input.

Local DB2 SQL Processing **For z/OS only**

Note that there are 2 activities related to DB2 on the ***WAS Application Definition*** window. Activity #8 should be used for local DB2 SQL processing, i.e. when DB2 is running on the same system image as WAS. This activity accounts for the CPU capacity requirement for DB2 SQL statement processing in the WAS application sizing estimate. This does not include the CPU capacity requirement for DB2 access using the JDBC/SQLJ connectors. Activity #9 should be used to account for the JDBC/SQLJ connector CPU capacity requirement for local and/or remote access to DB2. If your sizing includes Local DB2 SQL Processing then you should also include JDBC / SQLJ Access to Local DB2 which is part of activity #9 JDBC / SQLJ Access to DB2. However, when a WAS transaction, running under WAS on either Linux or z/OS accesses a remote DB2, running in another LPAR or on another processor, you should include JDBC / SQLJ Access to Remote DB2 which is part of activity #9 JDBC / SQLJ Access to DB2. Only the JDBC/SQLJ CPU processing that occurs in the WAS system image will be included in the capacity sizing estimate.

When you select Local DB2 SQL Processing, a window is displayed to rate your customer's DB2 SQL processing on a linear scale of 1 to 10. Move the slider to select a rating. Profiles of two common workloads used in WAS lab measurements are shown to help you select a rating. See DB2 Processing Workload Profiles for a description of the workloads. There is no default rating.

Defaults

There are no defaults for either Java Application Processing or Local DB2 SQL Processing.

For XML Document Processing, the default is 1 XML parse and 1 document created by a JSP or Java application. However, if the **Customize** button is used, XSL Transformation is another kind of XML processing that can be selected.

- XML Parsing: A simple 2k XML document is parsed using the SAX API. No XML validation is required during the parse. The simple XML document may be characterized as follows:
  35 elements per k of data

0 attributes per k of data

- Document creation by JSP or Java application default: A 2k XML document is created.
- No SSL protection unless SSL is included in the sizing. If SSL protection is included, all XML documents are protected.

Data Connectors to Other Applications

Description of Activities

zPSG provides sizing support for 4 data connectors for WAS applications running under z/OS: JDBC/SQLJ access to DB2, WOLA access to IMS, WOLA access to CICS, and JMS access to MQ. For WAS under Linux, we have performance data only for JDBC/SQLJ access to DB2 and JMS access to MQ.

JDBC / SQLJ Access to DB2

Description

This activity covers JDBC/SQLJ access from a WAS application (under either z/OS or Linux) to local and/or remote DB2.

If WAS is running under z/OS and DB2 is running in the same system image then select JDBC / SQLJ Access to Local DB2 . This activity only accounts for the CPU capacity requirement to access DB2 and does not account for DB2 SQL statement processing. In this case you should also select activity #8, Local DB2 SQL Processing, to account for the DB2 SQL processing CPU capacity requirement.

If WAS is running under z/OS in a different system image than DB2 then select JDBC / SQLJ Access to Remote DB2. If WAS is running under Linux, only JDBC / SQLJ Access to Remote DB2 is available. In this cases only the JDBC/SQLJ processing that occurs in the WAS system image is included in the estimate, the assumption being that the DB2 sizing might not be needed at all (if the DB2 processing already exists) or if needed, it would be done as a separate sizing exercise since it is running in a different system image or on a different processor.

However, although the DB2 SQL statement processing is not included, it does have an impact on the JDBC/SQLJ processing in the WAS system image, and so the approach used for characterizing DB2 SQL processing is also used also for local and remote DB2 access. You must select a rating on a scale of 1 to 10 to represent the amount and complexity of processing associated with local and/or remote DB2 access. When you select JDBC/SQLJ Access to DB2, a window pops up for rating your customer's DB2 connector processing for local and/or remote access on a linear scale of 1 to 10. Move the sliders to select a rating. Profiles of two common workloads used in WAS lab measurements are displayed using the **Application Profiles** button to help you select a rating. See DB2 Processing Workload Profiles for a description of the workloads. There is no default rating.

For local access to DB2 on z/OS, this activity assumes the use of the Type2 JDBC/SQLJ connector. For remote access to DB2 this activity allows for use of the new Type4 Universal connector or DB2 Connect. For both local and remote access to DB2, you can select either JDBC or SQLJ.

Defaults

- Complexity setting: No default, you must select a rating.
- Access Type: JDBC
- Connector: Type 2 for local access, Type 4 for remote access

Considerations
Note that the CPU cost for remote DB2 access is approximately 8% to 16% more than the CPU cost for local DB2 access when comparing the Type 4 connector for remote access to the Type 2 connector for local access. The CPU cost for SQLJ is approximately 10% less than JDBC. The CPU cost for DB2 Connect is approximately 5% more than the Type 4 connector.

WOLA Access to IMS — For z/OS only

Description
JCA (J2EE) connector support for access to IMS is provided by the WebSphere Optimized Local Adapters (WOLA). Data to be sent to IMS is generally stored in a COMMAREA, which is sent to IMS by the adapter. After the IMS transaction executes, the results are returned to WAS. The amount of data sent to and returned from IMS affects the CPU cost of adapter processing.

Defaults
The default is 1 call per transaction, 256 bytes are sent out to IMS, the same amount is always assumed to be returned.

Considerations/Explanation/Things to Think About
At the current time, we can provide sizing support only for access to IMS in the same LPAR as WAS. Adapter processing to IMS in another z/OS image would involve some additional capacity requirement. When IMS is running in the same LPAR as WAS, local mode is assumed.

WOLA Access to CICS — For z/OS only

Description
JCA (J2EE) connector support for access to CICS is provided by the WebSphere Optimized Local Adapters (WOLA). Data to be sent to CICS is generally stored in a COMMAREA, which is sent to CICS by the adapter. After the CICS transaction executes, the results are returned to WAS. The amount of data sent to and returned from CICS affects the CPU cost of adapter processing.

Defaults

The default is 1 call per transaction, 256 bytes are sent out to CICS, the same amount is always assumed to be returned.

Considerations/Explanation/Things to Think About
At the current time, we can provide sizing support only for access to CICS in the same LPAR as WAS. Adapter processing to CICS in another z/OS image would involve some additional capacity requirement. When CICS is running in the same LPAR as WAS, local mode is assumed. Note that CICS running in same z/OS image as WAS, and local mode, are required for 2 phase commit transactions.

JMS Back-end Connector to MQ

Description

This sizing support covers the use of JMS as a back-end data connector from a WAS application to WebSphere MQ. It is not for inbound JMS requests to WAS.

The CPU costs for MQGET and MQPUT processing are treated separately, so you can account for situations in which the WAS application sends a message to MQ (MQPUT) separately from receiving a message from MQ (MQGET). Note that use of JMS and MQ messaging implies an asymmetric programming model, so you can have situations involving only 1-way traffic.

There are two kinds of MQ messaging, point-to-point and publish/subscribe. Point-to-point messaging involves communication between one sender and one receiver for each message, and messages frequently (but not always) come in request/reply pairs. Publish/Subscribe messaging involves one sender and multiple receivers, who subscribe to a queue to receive these particular messages.

MQ messages also have several attributes, which are usually combined to achieve the desired level of CPU efficiency vs. message recoverability. There are 2 attributes for point-to-point messages:

- Non-persistent and non-transacted messages (also called express messages at times), which use less CPU but are not recoverable if the queue manager goes down
- Persistent and transacted messages, which are recoverable and involve a commit of the message to disk.

For publish/subscribe messages, the above attributes apply and there is an additional dimension:

- Non-durable vs. durable messages, which has an impact on how long messages stay in the queue waiting for receivers to receive them.

zPSG sizing support assumes that these attributes are always used in combination to make messages recoverable or not. Point-to-point messages are either non-persistent and non-transacted, or persistent and transacted. Publish/Subscribe messages are either non-persistent, non-transacted, and non-durable, or persistent, transacted, and durable.

Currently, Linux does not support Bindings Mode, only TCP/IP mode.

Defaults

Defaults are as follows:

- Point-to-Point messaging using TCP/IP mode
- 1 MQPUT plus 1 MQGET per transaction
- 2k nonpersistent, nontransacted messages (2 messages).
- If Publish/Subscribe messaging is selected, messages are nonpersistent, nontransacted, and nondurable.

Additional Input Fields Under Web Transaction Description

Peak transaction rate =

Description
Input field, numeric, required field, no range checking.
Enter the average transactions per second for your typical web transaction during a 15 minute interval of overall peak demand. See Web Transaction Description near the beginning of Help to decide what will be the scope of your web transaction. Be sure that the transaction rate appropriately matches the rate at which the sequence of activity that you have decided to call a transaction is done.

Default
No default, a transaction rate must be specified.

Considerations
As for most applications, WAS application capacity requirements are not directly related to the number of users but to the amount of work generated by users who are active. Therefore, the transactions executed per second is what capacity projections must be based on. In most cases, you will want to have enough processor capacity to support the transaction rate incurred during a 15 minute interval of peak demand. If you don't know a transaction rate, you may develop an estimated rate in one of the following ways:

- If you know the number of users to be supported:
  - Estimate the number of users who will submit at least 1 transaction during a 15 minute interval of peak demand.
  - Estimate the average number of transactions submitted by users in the 15 minutes.
  - Calculate the resulting number of transactions for the 15 minutes and divide by 900 sec.
- If you know the number of transactions per peak hour or per peak shift:
  - Calculate the number of transactions/sec.
  - Apply an average-to-peak ratio. This depends on the workload arrival patterns of the user population, i.e. does the transaction arrival rate tend to spike for short intervals (seconds) vs. be consistently high for longer intervals of time (minutes).
  - In general, the number of transactions per peak 15 minutes may be higher than for a peak hour, and even more so compared to a peak 8-10 hour time period. If you have statistics for a peak hour and you suspect that the rate for 15 minutes is higher, you might apply a factor of something like 1.1-1.25 to the hourly transaction rate. If you have statistics for a peak shift, you might apply a factor of 1.5:1 or 2:1, or something similar.

- If you know the average number of transactions per day or per week/month:
    - Calculate the number of transactions/sec
    - Apply an average-to-peak ratio (see the previous item for an explanation). A peak-to- average ratio to be applied to average transaction rate statistics for long periods of time like a day/week/month will probably be higher than a ratio for a peak hour to a peak 15 minutes. This ratio might be as high as 3:1 or 4:1, depending on the workload arrival patterns and “spikiness” of the user interaction with the website.

Reports and Capacity Projections

This section provides buttons to view output windows with summary reports and capacity projections.

Linux deployed as guest under z/VM (checkbox) **For Linux only**

When checked, the sizing will include capacity for z/VM and the **Linux under z/VM** button will be enabled. Note: the first time this checkbox is selected the ***Linux under z/VM*** definition window will be displayed.

Summary Report button

Click this button to view a summary of the input assumptions for the sizing and a breakdown of the CPU/transaction among the kinds of processing activity included in the sizing.

CPU Utilization button

Click this button to see an output window with estimates of processor capacity for all System z processors selected. A transaction rate target value must be input into the entry field to activate this button (a default value is not assigned).

Transaction Rate button

Click this button to see an output window with estimates of transaction rates that can be supported on all System z processors selected. You can also see the transaction rates that can be supported within a Saturation Design Point (SDP) specified for the processors.

SDP %

Description

Input field, numeric, valid range is 1 to 100.
SDP stands for Saturation Design Point. This is a classic capacity planning concept which allows you to examine the amount of workload than can be supported in less than the full capacity of the processor model. It applies to the Transaction Rate output window and enables you to determine how much work can fit into a processor that is already being used for other applications.

Default

The default is 90%.

Return button

Click this button to return to the ***Product Selection*** window.

<u>Reference-CPU</u> button

Click this button to go to a window to change the System z processor used as a basis for capacity ratings. See Reference-CPU for discussion concerning this setting.

<u>Linux under z/VM</u> button **For Linux only**

Click this button to define the z/VM environment when Linux is deployed as a guest under z/VM. See Linux under z/VM for details about these sizing considerations.

<u>Add to Aggregation</u> button

Click this button to add this workload to the aggregation candidate list.

Java Processing Workload Profiles

Below are Java processing profiles of two EJB 3.0 applications we used in lab performance studies. Application A consists of trivial transactions and is rated at 3 on a scale of 1 to 10. Application B consumes about 1.6 times as much CPU/transaction as Application A, and is rated at 5 on the scale. Although your application may differ in various ways from the activity described for these 2 workloads, use the profiles as a general guide and characterize the complexity and potential CPU consumption of your web application processing by rating it on a scale from 1 to 10.

Application A (DayTrader 2.0) Java application profile, rated 3 on the scale, metrics per transaction (average approximated):

- 1 inbound HTTP request to WAS
- 1 servlet invoked
- 1.5 JSPs invoked
- 3.0 stateless session bean invoked with Transaction_Required attribute
- 80 CMP entity beans invoked with Transaction_Required attribute
- Some transactions queue work to MQ
  - 0.1 MDB beans invoked with "Transaction Required" attribute
- No IIOP requests

Application B Java application profile, rated 5 on the scale, metrics per transaction (average approximated):

- 1 inbound HTTP requests and 2 inbound IIOP requests to WAS
- 1 servlets invoked
- 2 JSPs invoked
- 3.5 stateless session beans invoked, most with Transaction_Required attribute
- 9.3 CMP entity beans invoked with Transaction_Required attribute
- Some transactions queue work to MQ
  - 0.3 MDB beans invoked with Transaction_Required attribute

Metrics per transaction (average):

	Appl-A (DayTrader 2.0)	**Appl-B**
Number of IIOP requests	0	2.0
Number of servlets invoked	1.0	1.0
Number of JSPs invoked	1.5	2.0
Number of stateless session beans invoked	3.0	7.2
Transaction_Required	3.0	7.0
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	1.0
Number of stateful session beans invoked	0	0
Transaction_Required	0	0
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	0
Number of CMP entity beans invoked	80.4	9.3
Transaction_Required	80.4	9.3
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	0
Number of MDB beans invoked	0.1	0.3
Transaction_Required	0.1	0.3
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	0

DB2 Processing Workload Profiles

Below are DB2 processing profiles of two applications we use in lab performance studies.

Application A (DayTrader 2.0) consists of moderate DB2 processing and is rated at 3 on a scale of 1 to 10. It consists of:

- 4 to 8 simple SQL calls
- Mostly cursor selects
- On average, 1 update/insert/delete per 10 cursor selects

	# per Tran	Input Columns	Output Columns
Prepare	1.4		
Select	0	0	0
Insert	0.1	9.7	
Update searched	0.1	9.5	
Update cursor	0	0	
Delete searched	0	1	
Delete cursor	0	0	
Open	2.1	0.9	
Fetch	6.8		12.1

Ratio of Read-Only Commits to Update Commits:	7.1
% Hits in Dynamic Statement Cache:	99.9%

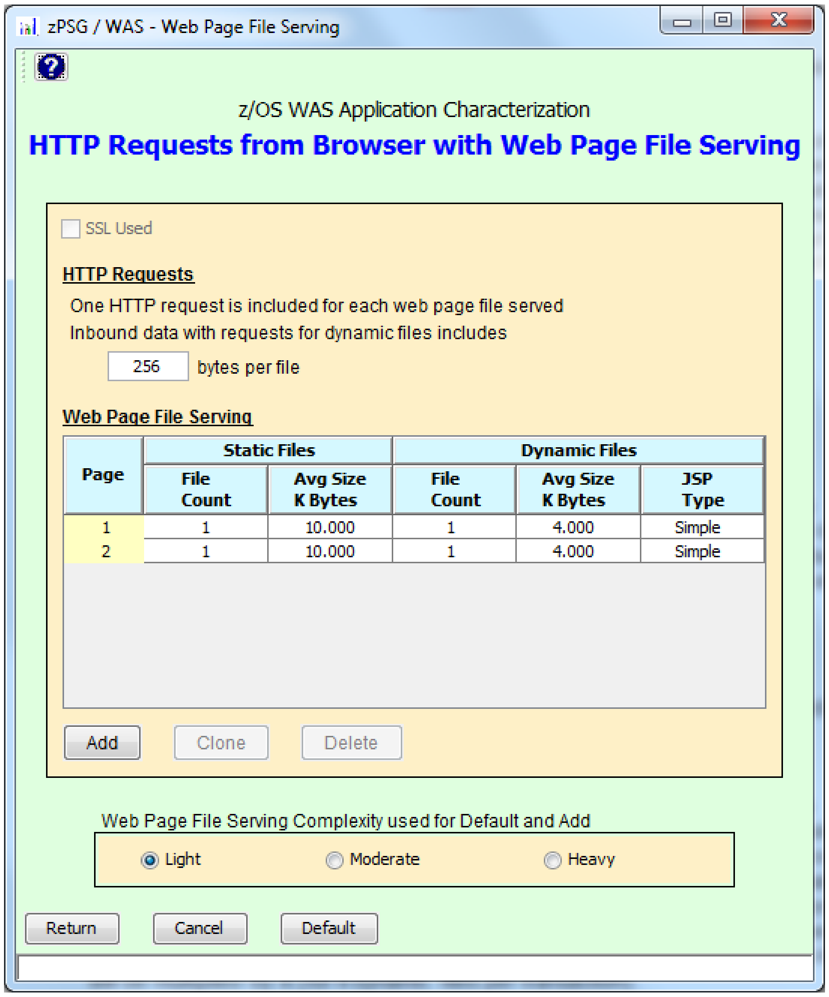
Application B consumes 3.5x more CPU per transaction than Application A, and is rated at 9 on the scale. It consists of:

- 11 to 14 simple SQL calls
- Relatively update intensive
- On average, 1 update/insert/delete per 3.5 cursor selects
- Uses XA global transaction and default JMS messaging

	# per Tran	Input Columns	Output Columns
Prepare	1.8		
Select	0	0	0
Insert	0.3	10.1	
Update searched	1.8	10.1	
Update cursor	0	0	
Delete searched	0.3	1	
Delete cursor	0	0	
Open	4.9	1.6	
Fetch	14.7		18.3

Ratio of Read-Only Commits to Update Commits: 1.0
% Hits in Dynamic Statement Cache: 99.9%

HTTP Requests from Browser with Web Page File Serving


zPSG / WAS - Web Page File Serving

z/OS WAS Application Characterization

HTTP Requests from Browser with Web Page File Serving

SSL Used [ ]

HTTP Requests

One HTTP request is included for each web page file served

Inbound data with requests for dynamic files includes 256 bytes per file

Web Page File Serving

Page	Static Files		Dynamic Files		
	File Count	Avg Size K Bytes	File Count	Avg Size K Bytes	JSP Type
1	1	10.000	1	4.000	Simple
2	1	10.000	1	4.000	Simple

Web Page File Serving Complexity used for Default and Add

[✓ Light] Moderate Heavy

This window is displayed when the **Customize** button for HTTP Requests from Browser with Web Page File Serving is clicked on the primary ***WAS Application Definition*** window.

Web pages are composed of static and dynamically created files by the browser. In general, simple web pages are composed of 1-5 files, medium pages of 10-15 files, and complex pages of 20-30 files. Generally, JSPs are invoked to format dynamic files.

There is usually a significant difference between the number of static files used by the browser to compose a web page and the number of static files served from System z. Many small frequently used static files (e.g. for buttons, logos, navigational aids) tend to be cached by the browser and are not served by a webserver (e.g. the IBM HTTP Server - IHS) or WAS during a typical web transaction. These static files **should not be included** in the number of static files served per web page.

In some System z WAS installations, static files are served by an edge server on another platform (along with DNS, SSL, and firewall functions) and not handled by IHS or WAS on z/Series; static files are sent by IHS or WAS to an edge server once, are cached there and served to browsers from that server.   These static files also **should not be included** in the number of static files served per web page.

In other cases, static files are served from IHS or WAS on System z as part of a typical web transaction (i.e. not cached in the browser or served from an edge server). They are the **only files** that should be included as part of the web file serving activity for a typical transaction. For sizing estimates, we are currently assuming that these files have been packaged in an EAR and are served via the WAS HTTP Transport Handler **from the web container**. We also assume they are cached by IHS or TCP/IP's FRCA and are not retrieved from HFS.

Description of Input Fields

Toolbar

? button   Click this button to go to Help for this window.

SSL Used (checkbox)
If checked, reflects that SSL is being used for protecting inbound HTTP data accompanying requests for dynamic web page files and outbound static and dynamic files that are served from WAS. If SSL protection is used in the transaction but not for protecting HTTP requests for web page files or serving web pages, you can uncheck this box.

HTTP Requests

One HTTP is automatically included for each web page file defined. No specific definition of these HTTP request is required.

bytes per file (input field)
In this field we provide the ability to specify the average size of inbound data accompanying requests for dynamic files. This is generally where input to WAS transactions comes into WAS. Specify the average size of inbound data accompanying **each request** for a dynamic file. Capacity to handle the receiving and translation of this amount of data (from ASCII to EBCDIC to Unicode) will be added for each dynamic file defined. For example, if 2 web pages are defined and each web page involves serving 2 dynamic files, CPU costs for handling the number of bytes specified in this input field will be multiplied by 4 (for 4 dynamic files per transaction).

Web Page File Serving

In this table, you can specify the number of web pages served per transaction, and the number and average size of static and dynamic files served per page. In the table, each web page served is represented by a row in the table. By default 2 pages are shown. Use the Add, Clone, or Delete buttons to change the number of web pages served per transaction. Within each row, specify the number and average size of both static and dynamic web page files served per page. Click on each field to change the number of files or average file size, specified in K bytes. Each dynamic file is assumed to be created by a JSP. Use the last column on the right to characterize JSP processing for the dynamic files on each page as Simple, Moderate, or Complex.

These buttons can be used to "populate" the file serving input for any web pages that you want to add to the table. Note that you can always over-ride the content of each row by double clicking on the field and entering a changed value for the number of static or dynamic files, the average file size, or the JSP complexity.

Add button
Click this button to add another web page to the table using one of 3 pre-defined levels of file serving activity. Select one of the 3 radio buttons near the bottom of the window to select Light, Medium, or Heavy file serving activity. See the profiles below for the number of static and dynamic files associated with each level of activity.

Clone button
If you want to add another web page with the same number of files and average file sizes as a page that is already in the table, highlight that row and click the **Clone** button to add another row (web page) with the same file serving input.

Delete button
If you want to delete a web page from the table, highlight the page and click the Delete button.

Web Page File Serving Complexity used for Default and Add (radio buttons):
Light, **Medium**, or **Heavy**
These buttons can be used to "populate" the file serving input for any web pages that you want to add to the table using the **Add** button, or change using the **Default** button. Use these buttons to select pre-defined levels of file serving activity for a web page. Here are descriptions of the file serving activity associated with each profile:

Light	1 static file of 10k and 1 dynamic file of 4k per web page, JSP processing to create dynamic file is Simple
Medium	7 static files of 10k each and 3 dynamic files of 4k each per web page, JSP processing to create dynamic files is Medium
Heavy	13 static files of 10k each and 5 dynamic files of 4k each per web page, JSP processing to create dynamic files is Complex.

Light file serving activity is the default. In most cases, this would cover the situation in which most static files used by the browser to compose a web page are cached on the browser or served from an edge server or proxy server rather the from the WAS in which the transaction runs.

Return button

Click this button to return to the ***WAS Application Definition*** window with changes you have entered.

Cancel button
Click this button to return to the ***WAS Application Definition*** window without saving any changes that you entered.

Default button
Click this button to restore defaults for this window.

Defaults

Defaults for **HTTP Requests from Browser with Web Page File Serving** are:

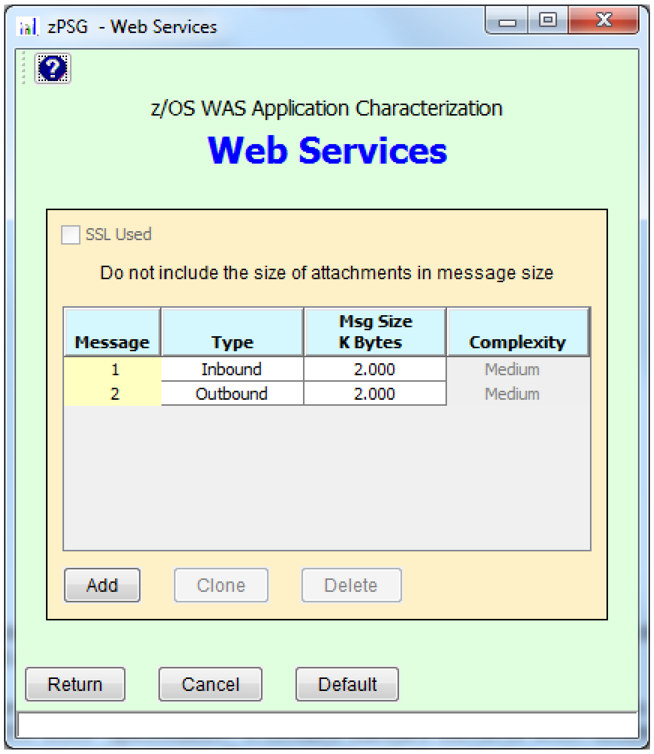
- 1 HTTP request is assumed for each web page file served (and this cannot be changed). Each request for dynamic files is accompanied by 256 bytes of inbound data.
- 2 web pages served per transaction
- Light web page file serving activity: 1 static file of 10k and 1 dynamic file of 4k with Simple JSP processing served per web page
- No SSL protection unless SSL is included on the ***WAS Application Definition*** window. If SSL is included, all inbound data and web page files are SSL protected.

Considerations

Our sizing support for receiving requests and serving web page files is currently based on interfacing with WAS via the WAS HTTP Transport Handler, not the separate IBM HTTP Server (IHS). While we don't have specific performance data on the IHS plug-in to WAS, we know that the pathlength is longer than the WAS Transport Handler for receiving requests into WAS and serving dynamic files, and shorter for serving static files.

The number of static web page files served from WAS is generally less than the number of files used by the browser to compose the web page. Many small, frequently used static files are cached by the browser and not served as part of a web transaction; other static files are cached on an Edge Server and not served from System z. Be sure not to over-state the number of static files served per transaction (especially if SSL protection is included), as this will significantly increase the capacity requirement.

<u>Web Services</u>



Do not include the size of attachments in message size

Message	Type	Msg Size K Bytes	Complexity
1	Inbound	2.000	Medium
2	Outbound	2.000	Medium

This window is displayed when the **<u>Customize</u>** button for <u>Web Services</u> is clicked on the ***WAS Application Definition*** window.

A web services request uses an XML document which conforms to the rules prescribed by the W3C SOAP specification. It contains standard elements which dictate the processing required to process the request as well as application unique content. Web services sizing support in **zPSG** is based on the JAX-WS web services specification available in current releases of WAS.

Web services messages are generally text but can contain other data formats such as encrypted data which may affect the amount of processing required. These other variations are not supported by **zPSG**. Web services messages may also contain an attachment. When this is the case, do not include the attachment in the size of an inbound message, since the attachment is not parsed.

Description of Input Fields

Toolbar

? **button** Click this button to go to Help for this window.

SSL Used (checkbox)
If checked, reflects that SSL is being used for protecting inbound and outbound web services messages. If SSL protection is used in the transaction but not for protecting web services messages, you can uncheck this box.

Message column in table
Entries for 2 messages are provided. If there are more than 2 messages in a typical transaction, use the **Add** or **Clone** buttons to add entries to the table.

Type column in table
Specify either **Inbound** or **Outbound** for each message.

Message Size column in table
Specify in Kbytes the average message size for each message (disregarding any attachments).

Complexity column in table
Currently this entry field is not active because we have performance data only for messages considered to be of medium complexity. If more performance data becomes available in the future, we will add the capability to designate Simple and Complex messages in addition to Medium.

Add button
Click here to add another message entry to the table.

Clone button
Click here to add another message to the table with the same values as the selected message entry.

Delete button
Click here to delete the selected message entry from the table.

Return button

Click this button to return to the *WAS Application Definition* window with changes you have entered.

Cancel button
Click this button to return to the *WAS Application Definition* window without saving any changes that you entered.

Default button
Click this button to restore defaults for this window.

Defaults
For the current web services sizing support, if web services is included in a web transaction, we assume the following:

- 1 web service request message of 2k and medium complexity is sent to WAS
- 1 web service response of 2k and medium complexity is returned to the client
- No SSL protection unless SSL is included in the sizing. If SSL protection is included, all web services messages are protected.

The web service request and response messages may be characterized as follows:

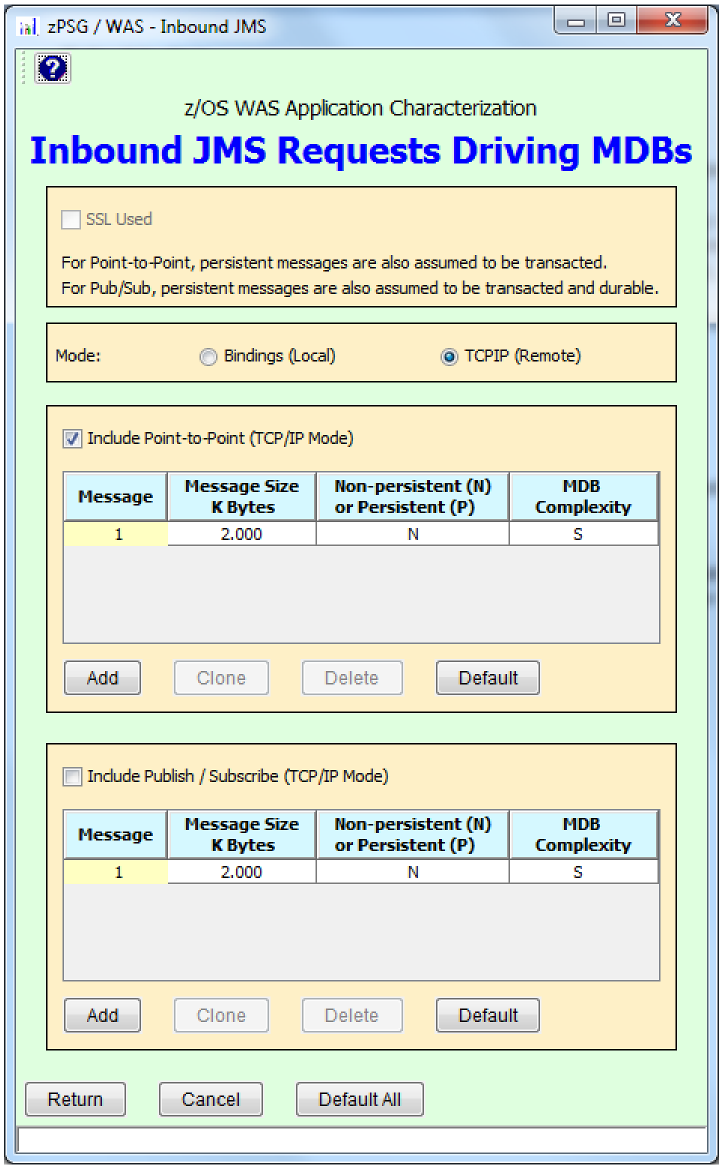
- Elements per K of data: 10
- Attributes per K of data: 0

General assumptions used for sizing

- The messages are text and do not include attachments.
- Default serializers and deserializers are used.
- CPU costs do not include application logic contained in the web service.

Considerations/Explanation/Things To Think About
The use of the name, SOAP, for both the messaging protocol standard and the original web services implementation is confusing. Both the old and the new web services engines comply with the SOAP standard.

Inbound JMS Requests Driving MDBs



z/OS WAS Application Characterization

Inbound JMS Requests Driving MDBs

☐ SSL Used

For Point-to-Point, persistent messages are also assumed to be transacted.
For Pub/Sub, persistent messages are also assumed to be transacted and durable.

Mode: ○ Bindings (Local) [✓ ● TCPIP (Remote)]

☑ Include Point-to-Point (TCP/IP Mode)

Message	Message Size K Bytes	Non-persistent (N) or Persistent (P)	MDB Complexity
1	2.000	N	S

☐ Include Publish / Subscribe (TCP/IP Mode)

Message	Message Size K Bytes	Non-persistent (N) or Persistent (P)	MDB Complexity
1	2.000	N	S

This window is displayed when the **Customize** button for Inbound JMS Requests Driving MDBs is clicked on the ***WAS Application Definition*** window.

This activity covers the use of JMS as an inbound request to WAS which executes a Message Driven Bean (MDB) to initiate or execute a transaction. It is not for JMS messages sent by or received by an active WAS transaction like a data connector function (which is covered by activity #12).

Description of Input Fields

Toolbar

? **button** Click this button to go to Help for this window.

SSL Used (checkbox)
If checked, reflects that SSL is being used for protecting inbound JMS messages that drive MDBs. If SSL protection is used in the transaction but not for protecting these JMS messages, you can uncheck this box.

There are 2 sections in this window, 1 for point-to-point messaging and 1 for publish/subscribe messaging. Both sections have the same input.

Mode selection buttons (z/OS only)
Select which mode is being used. Either TCP/IP (remote) or Bindings (local) mode.

Include (Point-to-Point or Publish/Subscribe) box
Check the box to include either point-to-point or publish/subscribe messages in the typical WAS transaction. Both boxes can be checked if you have both types of messages.

Message column in table
Not a direct input field. An entry for 1 message is provided. If there are additional messages in a typical transaction, use the **Add** or **Clone** buttons to add entries to the table.

Message Size column in table
Specify in Kbytes the average message size for each message.

Non-persistent or persistent column in table
Specify whether the message is **Non-persistent (N)** or **Persistent (P)**. For point-to-point messages, non-persistent also implies non-transacted, and persistent implies transacted. For publish/subscribe messages, non-persistent implies non-transacted and non-durable whereas persistent implies transacted and durable. Persistent, transacted (and durable) messages are recoverable if the MQ queue manager goes down. Non-persistent, etc. message are not recoverable.

MDB Complexity column in table
Specify whether the MDB processing is Simple, Medium, or Complex.

Add button
Click here to add another message entry to the table.

Clone button
Click here to add another message to the table with the same values as the selected message entry.

Delete button
Click here to delete the selected message entry from the table.

Default button
Click this button to restore defaults for the section of the window you are in, for either point-to-point or publish/subscribe messages.

Return button

Click this button to return to the ***WAS Application Definition*** window with changes you have entered.

Cancel button
Click this button to return to the ***WAS Application Definition*** window without saving any changes that you entered.

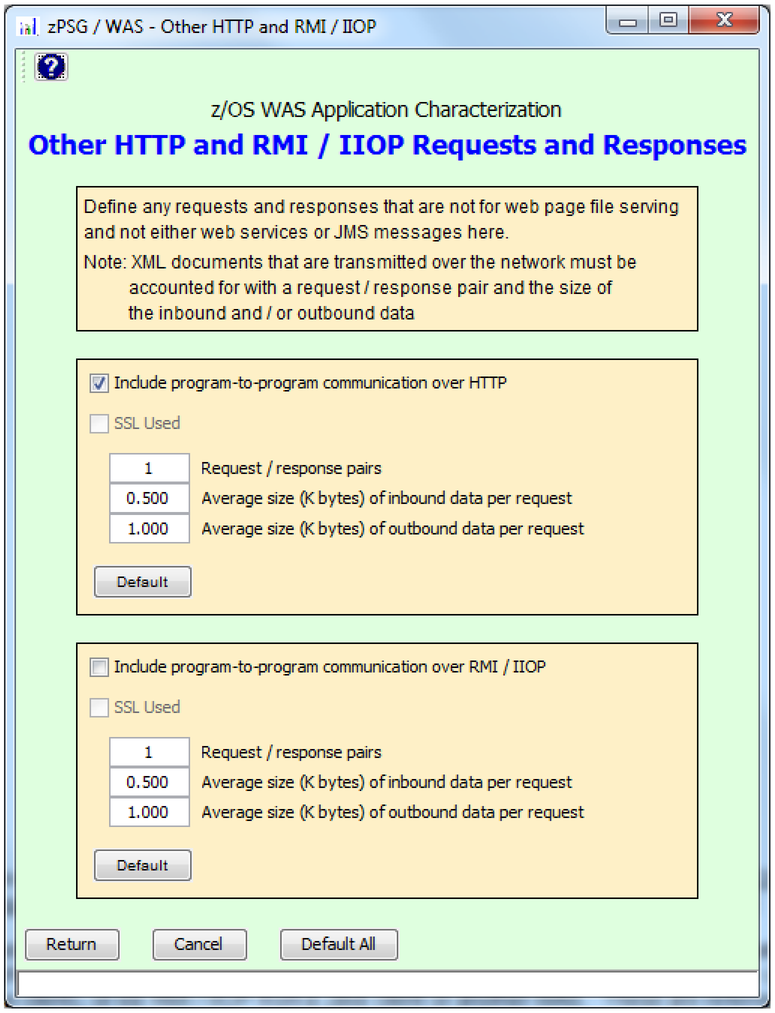
Default all button
Click this button to restore **all** the defaults for this window (for both point-to-point and publish/subscribe messages).

Defaults
Overall default is point-to-point messaging. Other defaults:

- TCP/IP (remote) mode
- 1 inbound request per transaction
- 2k non-persistent (and non-transacted) message
- MDB complexity is Simple
- If publish/subscribe messaging is included, same defaults apply for number & size of message and MDB complexity.
- If SSL is included, all inbound JMS messages are protected.

Considerations
The use of JMS and MQ messaging implies an asymmetric programming model, so this activity includes a request to WAS, but not a response. If there is a response from the WAS transaction, you must define it separately. It could be a web page that is served, some outbound data transmitted via HTTP or RMI/IIOP (use activity #4 with outbound data for this type of response), or any other type of response. Be sure to include consideration of a response from the transaction if there is one.

Other HTTP and RMI / IIOP Requests and Responses



z/OS WAS Application Characterization

Other HTTP and RMI / IIOP Requests and Responses

Define any requests and responses that are not for web page file serving and not either web services or JMS messages here.

Note: XML documents that are transmitted over the network must be accounted for with a request / response pair and the size of the inbound and / or outbound data

[✓] Include program-to-program communication over HTTP

[ ] SSL Used

1	Request / response pairs
0.500	Average size (K bytes) of inbound data per request
1.000	Average size (K bytes) of outbound data per request

[ ] Include program-to-program communication over RMI / IIOP

[ ] SSL Used

1	Request / response pairs
0.500	Average size (K bytes) of inbound data per request
1.000	Average size (K bytes) of outbound data per request

This window is displayed when the **Customize** button for Other HTTP and RMI / IIOP Requests and Responses is clicked on the ***WAS Application Definition*** window.

In addition to or instead of HTTP requests for web page files, the customer's application might involve requests coming to WAS via HTTP from another application program (a Java client), or via RMI / IIOP from a Java client or another WAS. These are referred to as program-to-program communication requests (as distinct from HTTP requests from a browser for web page files). Typically, this communication involves both a request (perhaps with inbound data) and a response (with outbound data), although 1-way communication is possible.

If you are sizing a significant WAS application to run behind WebSphere Portal Server, you can use these request/response pairs if appropriate for communication between the portlet and the rest of the WAS application.

As noted on the input screen, program-to-program communication should be used to account for transmission costs for XML documents that go over the network. Be sure to include an HTTP or RMI/IIOP request/response pair for any XML documents, with suitable data sizes.

Description of Input Fields
Toolbar
? button Click this button to go to Help for this window.

There are 2 sections on this window, one for HTTP and one for RMI/IIOP requests/responses. You may have one or both in a typical transaction.

Include program-to-program communication over HTTP or **RMI/IIOP** (checkboxes).
Check one or both of these boxes to included a request to WAS that is not associated with either web page file serving, web services messages, or inbound JMS messages that drive MDBs. If you have (non-web services) XML documents that are transmitted over the network, use the appropriate box to include the request associated with the XML document(s). Use the inbound/outbound data fields below to account for transmission and conversion CPU costs of the XML documents.

SSL Used (checkbox)
If checked, reflects that SSL is being used for protecting inbound and outbound data accompanying HTTP or RMI/IIOP requests (other than for web page file serving). If SSL protection is used in the transaction but not for protecting this HTTP or RMI/IIOP communication, you can uncheck this box.

Request / response pairs (numeric input field)
Enter the number of request/response pairs per transaction. Typically there would be both inbound data coming in with the request and outbound data going out with the response.

Average size (K bytes) of inbound data per request (numeric input field)
Enter the average size in K bytes of inbound data per HTTP or RMI/IIOP request. If you have multiple requests with different amounts of data, compute the average size. For example, you might have 2 requests, one accompanied by 512 bytes of data, and another request with a 2k XML document. The average size would be 0.5k + 2k = 2.5k / 2 = 1.25k. If you have a situation in which the communication is in one direction only (either inbound or outbound), enter "0" for the data size for the opposite direction.

Average size (K bytes) of outbound data per request (numeric input field)
Enter the average size in K bytes of outbound data per HTTP or RMI/IIOP request. If you have multiple responses with different amounts of data, compute the average size. For example, you might have 2 responses, one with 1k of data, and a separate outbound 2k XML document. The average size would be 1k + 2k = 3k / 2 = 1.5k. If you have a situation in which the communication is in one direction only (either inbound or outbound), enter "0" for the data size for the opposite direction.

Default button
Click this button to restore defaults for either the HTTP or the RMI/IIOP section of this window.

Return button

Click this button to return to the *WAS Application Definition* window with changes you have entered.

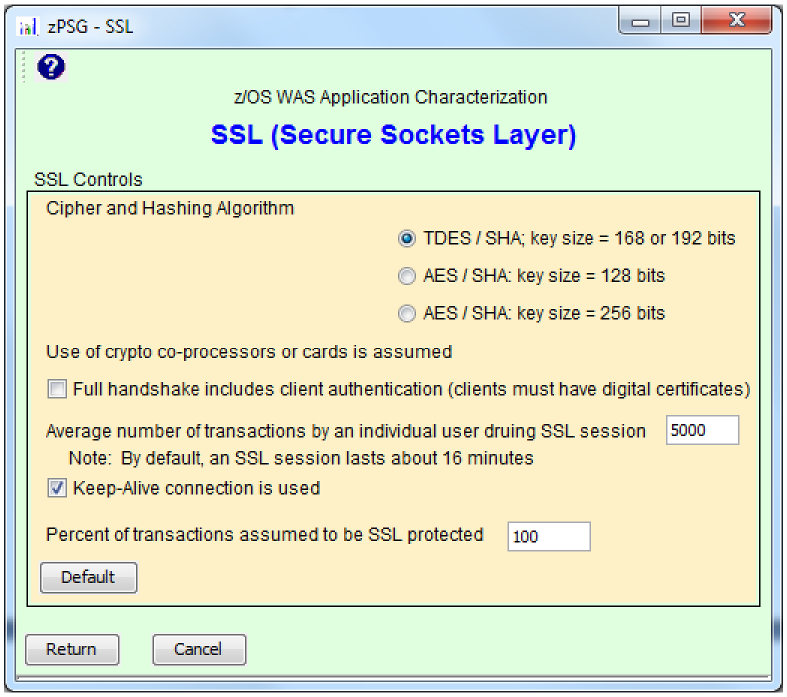
Cancel button
Click this button to return to the *WAS Application Definition* window without saving any changes that you entered.

Default All button
Click this button to restore defaults for the entire window (for both HTTP and RMI/IIOP communication).

Defaults
The default is 1 HTTP request/response pair, consisting of an HTTP request with 0.5k of inbound data and 1k of outbound data as a response. If RMI/IIOP is selected, the data size defaults are the same, but RMI/IIOP communication is not selected by default.

Considerations
RMI / IIOP request CPU costs are somewhat lower than HTTP request costs for small amounts of data. We don't currently have performance data for the CPU costs associated with transmitting data via RMI/IIOP so we currently use the CPU costs for HTTP data for both communications protocols. **Be aware, however, that with RMI/IIOP communication CPU costs can escalate significantly when large objects are transmitted, and our current sizing support cannot take these situations into account.**

WAS SSL



z/OS WAS Application Characterization

SSL (Secure Sockets Layer)

SSL Controls

Cipher and Hashing Algorithm

- [✓ TDES / SHA; key size = 168 or 192 bits]
- AES / SHA: key size = 128 bits
- AES / SHA: key size = 256 bits

Use of crypto co-processors or cards is assumed

☐ Full handshake includes client authentication (clients must have digital certificates)

Average number of transactions by an individual user druing SSL session: 5000

Note: By default, an SSL session lasts about 16 minutes

☑ Keep-Alive connection is used

Percent of transactions assumed to be SSL protected: 100

This window is displayed when the **Customize** button for SSL is clicked on the primary ***WAS Application Definition*** window.

SSL (Secure Sockets Layer) sizing support is based on JSSE for WAS on z/OS and Linux.

zPSG supports 2 of the several SSL ciphers/hashing algorithms available, mostly because the others are not widely used. The most popular cipher is TripleDES/SHA (TDES). TDES, used by banks, financial institutions and government agencies requiring very high levels of encryption security, is supported by crypto assist hardware, for both the full SSL handshake and encryption/decryption of data. The AES/SHA cipher/hashing algorithm is also supported. You would not want to use TDES or AES without crypto hardware.

Note that another cipher/hashing algorithm that is supported by IBM crypto hardware, DES/SHA, is not included in **zPSG** since it is not widely used.

Description of Input

SSL Controls

Cipher and Hashing Algorithm (radio buttons):

- **TDES/SHA**; key size = 168 or 192 bits
- **AES/SHA**; key size = 128 bits
- **AES/SHA**; key size = 256 bits

Click the appropriate radio button to select the cipher you want. Note that TDES/SHA or AES/SHA should not be used without configuring crypto cards (or using crypto co-processors on some older zSeries models).

Use of crypto co-processors or cards is assumed.
This is not an input field; it is simply an informational message documenting that our sizing support for SSL assumes that whenever SSL processing can be offloaded from general purpose CPs on System z processors to either crypto co-processors (on older zSeries models) or crypto cards (on newer models), it is offloaded. For TDES/SHA and AES/SHA, this assumption is significant because both full handshake processing and data encryption/decryption processing are offloaded. The CPU capacity requirement for TDES/SHA and AES/SHA SSL protection would be far greater if crypto cards were not available.

Full handshake includes client authentication (checkbox)

If checked, the cost of a full handshake with client authentication is included in the sizing. This cost is quite a lot higher than the cost of a full handshake, but the impact on a sizing estimate depends on how frequently full handshakes are required. Client authentication processing does not occur with cached handshakes. If the number of WAS transactions by an individual user during an SSL session (see below) is high, the portion of handshakes that are full handshakes vs. cached handshakes is reduced and therefore the impact of client authentication is lower.

Average number of transactions by an individual user during SSL session

This input is used to control how our tool pro-rates the CPU cost of handshakes for a transaction. We compute the number of handshakes required per transaction based on what is being SSL protected. SSL works as follows: the first handshake in an SSL session is a full handshake (perhaps with client authentication). Subsequent handshakes in the SSL session, even for new WAS transactions, are cached handshakes. Based on the number entered in this field, we pro-rate the CPU cost for handshakes between full and cached handshakes. Since cached handshakes use significantly less CPU than full handshakes, the higher the number of transactions during an SSL session, the lower the pro-rated handshake cost. As noted on the screen, by default an SSL session lasts approximately 16 minutes, although it can be changed by parameter. For sizing purposes, we don't care how long your SSL session lasts as long as the value in this field correctly reflects the average number of transactions done by individual users during the session.

Keep-Alive Connection is used

Select this option when the socket connection between the client and the WAS server is kept open long enough to process multiple SSL requests/responses. Doing so reduces the number of cached handshakes used in an SSL session. By default this option is selected.

Percent of transactions assumed to be SSL protected

You can use this field to reflect situations in which only some WAS transactions involved SSL protection. In most cases, the default of 100% will apply.

Push Buttons

Default button

Click this button to restore defaults for the SSL Controls section of the window.

Return button

Click this button to return to the ***WAS Application Definition*** window with changes you have entered.

Cancel button

Click this button to return to the ***WAS Application Definition*** window without saving any changes that you entered.

Defaults

If SSL is selected on the WAS Application Definition screen, the default is that all inbound & outbound data, messages, files, documents, etc. are protected.

- The default cipher and hashing algorithm is TDES/SHA.
- Full handshake does not involve client authentication.
- Individual users do 5000 transactions within an SSL session.
- Keep-Alive Connection is used.
- SSL protection is used for 100% of the transactions.

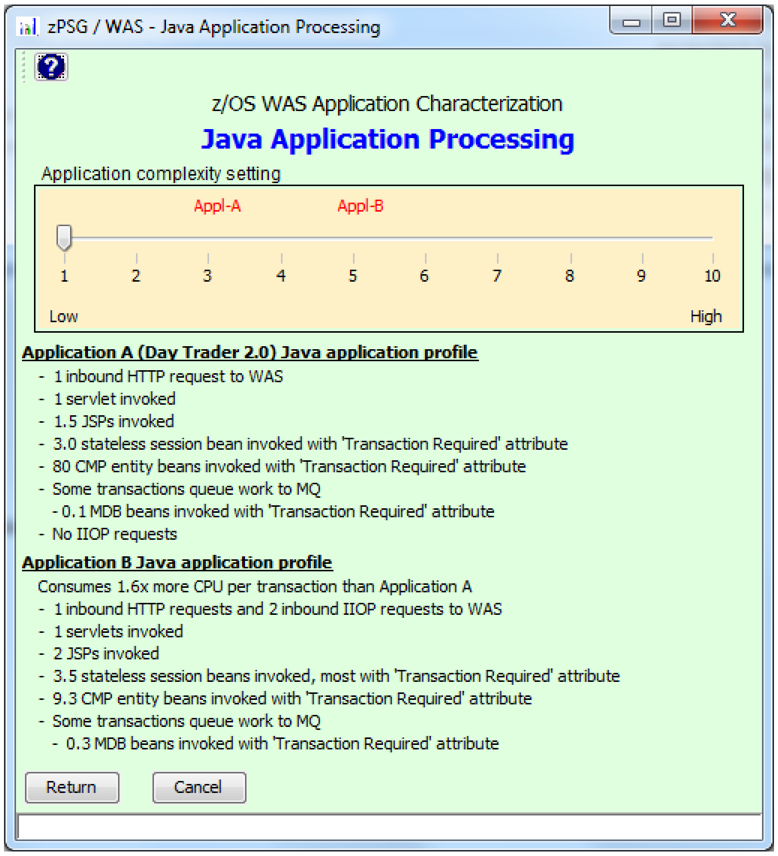
Sizing Assumptions
Crypto assist co-processors and cards are available (used for full handshakes and, for TDES and AES, for encryption/decryption).
1 handshake per web page served
1 handshake for JMS or web services messaging
1 handshake per request/response pair for other HTTP or RMI/IIOP communication
Keep-alive connection assumes 100 SSL requests/responses per socket connection.

Considerations
Decisions about the use of SSL, including the choice of cipher, are based on the level of security required for an application. Data transfers involving financial payments or sensitive government information are usually protected by TDES/SHA. When this is the situation, be sure to include crypto hardware in the solution because CPU costs for TDES and AES without crypto assist are very high. Our sizing support for SSL assumes the use of crypto hardware assist.

Full SSL handshakes, even when supported by crypto hardware consume significant amounts of CPU capacity. Therefore, if the customer's users tend to do a lot of transactions, one right after another for a sustained period of time (e.g. bank tellers), it would be a good idea to set the SSL session timeout parameter to a fairly long time in order to reduce full SSL handshakes. Cached handshakes, which can be used after the first handshake in an SSL session, use significantly less CPU. To make the WAS sizing estimate correctly represent capacity requirements for applications involving lots of SSL protected data, be sure that the **Average number of transactions by an individual user during SSL session** input accurately reflects the customer's situation and the **Keep-alive connection** option is selected appropriately.

If you are trying to compare WAS sizing estimates for System z with WAS estimates for other platforms, be aware that there is a significant difference in how SSL is treated in their WAS sizing estimates. Their sizing tool increases the CPU/tran by a certain percentage to account for SSL. System z SSL sizing support is based on performance data for protecting different amounts of data for two different ciphers, and yields much more accurate capacity estimates for using SSL. In our sizing support SSL can account for anywhere from 5% to 50% of the CPU/tran. If your customer has relatively trivial Java (and DB2 if applicable) application processing, but large amounts of data that is SSL protected, the CPU costs for SSL will be a high percentage of the CPU/tran. If they have significant Java application processing and very little data being protected, the percentage could be low. So you cannot create an "apples-to-apples" comparison of the capacity on System z vs. other platforms for a WAS application, especially when SSL is involved.

Java Application Processing



z/OS WAS Application Characterization

Java Application Processing

Application complexity setting

Appl-A Appl-B

1 2 3 4 5 6 7 8 9 10

Low High

Application A (Day Trader 2.0) Java application profile

- 1 inbound HTTP request to WAS
- 1 servlet invoked
- 1.5 JSPs invoked
- 3.0 stateless session bean invoked with 'Transaction Required' attribute
- 80 CMP entity beans invoked with 'Transaction Required' attribute
- Some transactions queue work to MQ
  - 0.1 MDB beans invoked with 'Transaction Required' attribute
- No IIOP requests

Application B Java application profile

Consumes 1.6x more CPU per transaction than Application A

- 1 inbound HTTP requests and 2 inbound IIOP requests to WAS
- 1 servlets invoked
- 2 JSPs invoked
- 3.5 stateless session beans invoked, most with 'Transaction Required' attribute
- 9.3 CMP entity beans invoked with 'Transaction Required' attribute
- Some transactions queue work to MQ
  - 0.3 MDB beans invoked with 'Transaction Required' attribute

This window is displayed when the **Customize** button for Java Application Processing is clicked on the primary ***WAS Application Definition*** window.

This window is for characterizing the amount and complexity of the transaction logic, or business processing, of a typical WAS transaction. The CPU capacity requirement of a Java application depends on a number of factors, including the number of servlets, JSPs and EJBs executed, whether session is stateful or stateless, and the processing complexity of the Java methods. The amount of data processed also has an impact on capacity.

Characterizing your customer's typical transaction by rating the Java processing on a linear scale of 1 to 10. Profiles of the Java processing for 2 WAS applications used in lab performance measurements are provided to help you select a rating.

Data connector processing for CICS/IMS access is addressed separately. For Java processing, if your web transaction does not include much business logic in Java and is mainly a vehicle for sending a transaction to CICS or IMS, use a Java rating of 1.

Description of Input Fields

<u>Toolbar</u>

<u>?</u> button Click this button to go to Help for this window.

<u>Application Complexity Setting</u>

<u>Scale of 1 to 10 with Slider</u>

Move the slider by clicking on it with your cursor and dragging it to the rating you desire. The slider is initially set at a rating of 1 on the left side of the scale, but this should not be considered a reasonable common setting to default to when you aren't sure what rating to select. We have no way of knowing what your customer's application logic is like. To come up with a reasonable sizing estimate, you must work with your customer to select an appropriate rating.

Below the rating scale is Java processing profiles of 2 applications we used in lab performance studies. Application A consists of trivial transactions and is rated at 3 on a scale of 1 to 10. Application B consumes about 1.6 times more CPU per transaction as Application A, and is rated at 5 on the scale. Although your application may differ in various ways from the activity described for these 2 workloads, use the profiles as a general guide and characterize the complexity and potential CPU consumption of your web application processing by rating it on a scale from 1 to 10. Do not include any consideration of back-end processing here; it will be characterized separately. Include only the application processing performed in the WAS container, and DB2 data connector processing in an entity bean.

Application A (DayTrader 2.0) Java application profile, rated 3 on the scale, metrics per transaction (average approximated):

- 1 inbound HTTP request to WAS
- 1 servlet invoked
- 1.5 JSPs invoked
- 3.0 stateless session bean invoked with Transaction_Required attribute
- 80 CMP entity beans invoked with Transaction_Required attribute
- Some transactions queue work to MQ
  - 0.1 MDB beans invoked with "Transaction Required" attribute
- No IIOP requests

Application B Java application profile, rating 5 on the scale, metrics per transaction (average approximated):

- 1 inbound HTTP requests and 2 inbound IIOP requests to WAS
- 1 servlet invoked
- 2 JSPs invoked
- 3.5 stateless session beans invoked, most with Transaction_Required attribute
- 9.3 CMP entity beans invoked with Transaction_Required attribute
- Some transactions queue work to MQ
  - 0.3 MDB beans invoked with Transaction_Required attribute

Metrics per transaction (average):

	Appl-A (DayTrader 2.0)	**Appl-B**
Number of IIOP requests	0	2.0
Number of servlets invoked	1.0	1.0
Number of JSPs invoked	1.5	2.0
Number of stateless session beans invoked	3.0	7.2
Transaction_Required	3.0	7.0
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	1.0
Number of stateful session beans invoked	0	0
Transaction_Required	0	0
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	0
Number of CMP entity beans invoked	80.4	9.3
Transaction_Required	80.4	9.3
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	0
Number of MDB beans invoked	0.1	0.3
Transaction_Required	0.1	0.3
Transaction_Supported	0	0
Transaction_Requires_New	0	0
Transaction_Not_Supported	0	0

Push Buttons

Click the **Return** button to return to the primary ***WAS Application Definition*** window with any rating you select by moving the slider with your cursor.

Click the **Cancel** button to return to the primary ***WAS Application Definition*** window without saving the change in the rating. If you use this button instead of the Return button, the output windows will not reflect the Java rating the slider is left on.

Default

No default, select a rating on the scale of 1 to 10.

Considerations

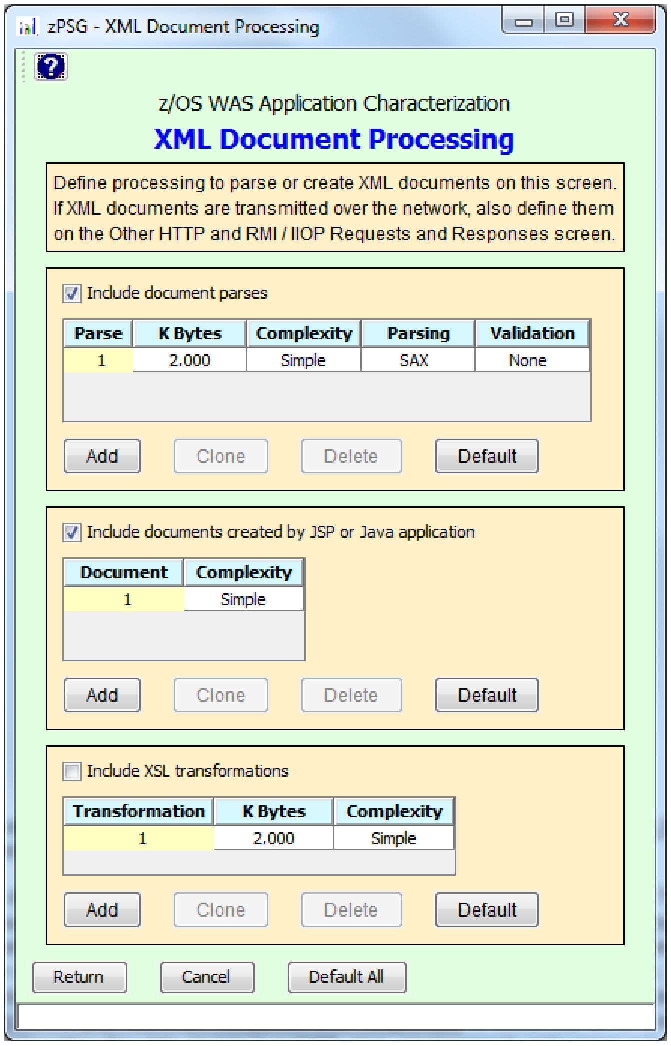
This scale represents the complexity of your WAS application logic. It does not include the cost of sending requests to WAS. It also does not include the cost of calling connectors to CICS or IMS, DB2, or the cost of DB2 SQL processing. Here are some things that might cause you to select a higher or lower Java application profile rating:

- If what you consider a transaction actually consists of several WAS requests, you should give your application profile a higher rating. If what you consider a transaction consists of a single request to WAS or a small number of WAS requests, you should give your application profile a lower rating.
- If your business logic is complex and it is performed in Java under WAS, you should give your application profile a higher rating. If your business logic is simple, or if you simply call CICS, IMS, or DB2 and perform your business logic there, you should give your application profile a lower rating. In many cases, a web transaction that is used to web-enable a CICS or IMS transaction and has little or no application processing under WAS and should be given a rating of 1.
- If you perform lots of validity checking or information parsing in your WAS application, you should give your application profile a higher rating. If you do not perform validity checking or do lots of information parsing in your application, you should give your application profile a lower rating.

The two model applications, Application A and Application B, represent two points of reference on the complexity scale. Information in the tables for these applications (above) should give you a feel for how complex they are relative to your application. This will help you select the right place for your application on the scale.

If you are trying to compare the capacity requirement for a WAS application on System z vs. other platforms, be aware that there is no consistency in characterizing the relative complexity or amount of application processing across platforms. What might be characterized as “complex” on another platform is likely to be seen as “moderate” on System z. It is extremely difficult to create an “apples-to-apples” comparison of WAS sizing estimates. There is currently no consistency for workload characterization, data sizes, basic sizing assumptions, nor lab performance measurement methodology for producing CPU cost data on which sizing is based. In general, System z WAS sizing estimates will be higher, but more realistic, than many of the estimates for other platforms.

XML Document Processing



z/OS WAS Application Characterization

XML Document Processing

Define processing to parse or create XML documents on this screen. If XML documents are transmitted over the network, also define them on the Other HTTP and RMI / IIOP Requests and Responses screen.

☑ Include document parses

Parse	K Bytes	Complexity	Parsing	Validation
1	2.000	Simple	SAX	None

☑ Include documents created by JSP or Java application

Document	Complexity
1	Simple

☐ Include XSL transformations

Transformation	K Bytes	Complexity
1	2.000	Simple

This window is displayed when the **Customize** button for XML Document Processing is clicked on the primary ***WAS Application Definition*** window.

XML documents are generally used for inter-application communication. They can be received by WAS with HTTP requests, sent by WAS to a client, or not transmitted over the network at all but received from or passed to another application running in the same system image. If XML documents are transmitted over the network, you will need to include the size of the documents as inbound and/or outbound data under activity #4, Other HTTP and RMI/IIOP Requests and Responses to cover the CPU cost of receiving and converting the data to run in a WAS Java environment or to send out an outbound document. You may also have an application in which XML processing occurs for documents that are not sent over the network. Inbound documents to WAS may or may not involve XML parsing. and parsing may or may not involve validation. Outbound documents may be created by JSPs or by XSL Transformation, or they may be created from a DOM tree. They could also be retrieved from DB2 or received back from CICS or IMS.

XML processing varies greatly depending on factors like the size and complexity of the document, whether parsing and/or validation are included, etc. The number of attributes per k of data is an important factor in determining the complexity of an XML document when using the SAX parser.

Note: Web services messages include XML documents but should be defined in the Web Services section. Do not describe web services messages as XML documents for sizing input.

Description of Input Fields

Toolbar

? button Click this button to go to Help for this window.

There are 3 sections to the XML input window. Each section can be included separately in a sizing estimate. The first section is for XML document parsing. The second section is for creating XML documents using a Java servlet or JSP. The third section is for creating XML documents via XSL Transformation. In each section, you should provide input on the number of times each of these activities is done per transaction, not necessarily on the number of documents being processed. For example, if a document is parsed twice during a transaction, there are 2 parses per tran even through there is only one document.

Include boxes for parsing, document creation by Java program, or by XSLT

Check here to include this type of XML processing in the transaction.

Table for XML Parsing

Use a row in the table for each parse (whether or not a separate document is involved). You can change the content of any input field in the table by double-clicking on the cell. Note that if any of the documents being parsed are also transmitted over the network, you must define them (once for each document transmission, regardless of how many times the document is parsed) on the ***Other HTTP and RMI/IIOP Requests and Responses*** window.

K Bytes column

Specify the size of the document being parsed in k.

Complexity column

Select the complexity of the parse, taking into account the number of elements and/or attributes and other factors that contribute to parsing complexity.

Parsing column

Specify whether SAX or DOM parsing is used.

Validation column

Select whether there is no validation, DTD validation, or Schema validation for this parse. Note that validation tends to use considerable CPU, especially Schema validation.

Table for Document Creation by Java Servlet or JSP

Use a row for each document creation by a Java program (not including XSLT processing). You can change the content of any input field in the table by double-clicking on the cell. Note that if the document created here is sent over the network, you must describe the document size on the ***Other HTTP and RMI/IIOP Requests and Responses*** window.

Complexity column

Select the complexity of the Java servlet or JSP. Note that document size is not requested in this window because data size is not necessarily related to servlet/JSP processing complexity. But, as mentioned above, document size must be specified on the ***Other HTTP and RMI/IIOP Requests and Responses*** window if the document is transmitted over the network.

Table for Document Creation by XSL Transformation

K Bytes column

Specify the size of the document being created by XSLT in k.

Complexity column

Select the complexity of XSLT processing, taking into account the number of elements and/or attributes and other factors that contribute to XSLT processing complexity.

Push Buttons for Each Section

Add button
Click here to add another parse or document creation entry to the table.

Clone button
Click here to add another parse or document creation to the table with the same values as the selected entry.

Delete button
Click here to delete the selected parse or document creation entry from the table.

Default button

Click this button to restore the defaults for the individual section of the window.

Push Buttons for Entire Window

<u>Return</u> button

Click this button to return to the ***WAS Application Definition*** window with changes you have entered.

<u>Cancel</u> button
Click this button to return to the ***WAS Application Definition*** window without saving any changes that you entered.

<u>Default all</u> button
Click this button to restore defaults for all three sections of the window.

Defaults

If XML Document Processing is included in the sizing, 1 XML parse and 1 document created by a JSP or Java application are included. However, if the customize button is used, XSL Transformation is another kind of XML processing that can be selected. Following are the defaults for each section

- <u>XML Parsing</u> default

  A simple 2k XML document is parsed using the SAX API. No XML validation is required during the parse. The simple XML document may be characterized as follows:
  35 elements per k of data
  0 attributes per k of data

- <u>Document creation by JSP or Java application</u> default

  An XML document of moderate size (about 2k) is created using a JSP or Java servlet with Simple processing. Although this is not really XML processing, it is included here because it involves an XML document.

- <u>XSL Transformation</u> default

  If XSLT is included, a 2k XML document is created via Simple XSLT processing (by default, XSLT is not included when XML Processing is included on the ***WAS Application Definition*** window – it must be "turned on in this window".

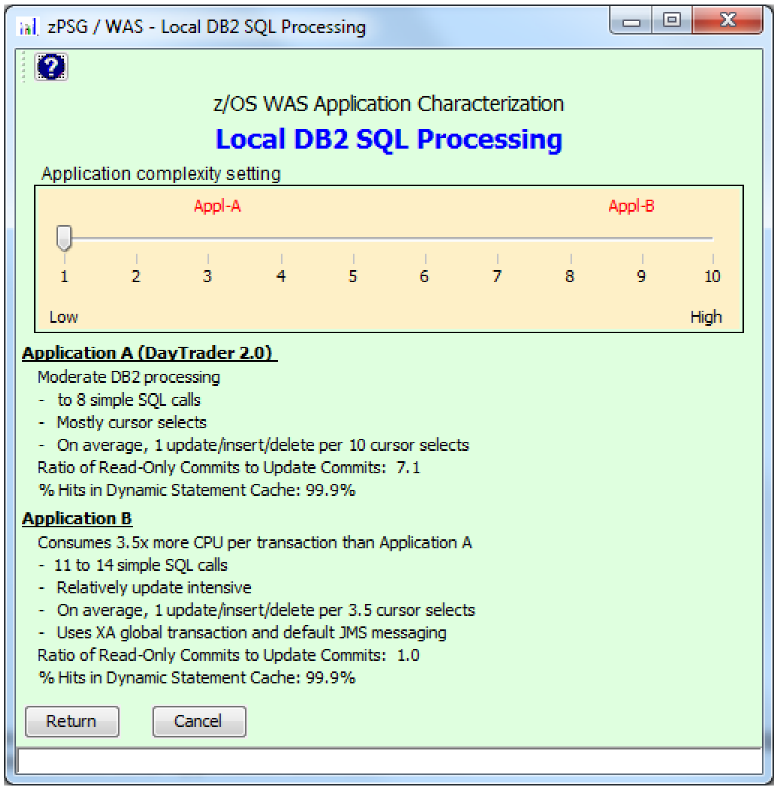
- No <u>SSL protection</u> unless SSL is included in the sizing. If SSL protection is included, all XML documents are protected.

General assumptions used for sizing

- XML4J 4.3 parser from the XML Toolkit for z/OS and OS390 (based on Xerces 2.6.2)

- Parsing CPU costs do not include application logic required to handle document contents.

Local DB2 SQL Processing



z/OS WAS Application Characterization

Local DB2 SQL Processing

Application complexity setting

Application A (DayTrader 2.0)

Moderate DB2 processing
- to 8 simple SQL calls
- Mostly cursor selects
- On average, 1 update/insert/delete per 10 cursor selects

Ratio of Read-Only Commits to Update Commits: 7.1

% Hits in Dynamic Statement Cache: 99.9%

Application B

Consumes 3.5x more CPU per transaction than Application A
- 11 to 14 simple SQL calls
- Relatively update intensive
- On average, 1 update/insert/delete per 3.5 cursor selects
- Uses XA global transaction and default JMS messaging

Ratio of Read-Only Commits to Update Commits: 1.0

% Hits in Dynamic Statement Cache: 99.9%

This window is displayed when the **Customize** button for Local DB2 SQL Processing is clicked on the ***WAS Application Definition*** window.

The CPU capacity requirement for DB2 SQL statement processing depends on the number and type of SQL statements (Select, Insert, Update, or Delete), the volume of data processed by DB2, the number of rows returned by DB2 to the application, whether the application is executing local or remote to the DB2 server, and the complexity of any stored procedures. This window is for characterizing the DB2 SQLJ statement processing for a WAS transaction that accesses a **local** DB2, running in the same z/OS LPAR as WAS. Only the CPU costs of the DB2 SQL statement processing is included in the estimate for this activity. To account for JDBC / SQLJ access to DB2, select activity #9 on the ***WAS Application Definition*** window and include JDBC / SQLJ Access to Local DB2.

Description of Input Fields

Toolbar

? button Click this button to go to Help for this window.

Application Complexity Setting

Scale of 1 to 10 with Slider

Move the slider by clicking on it with your cursor and dragging it to the rating you desire. The slider is initially set at a rating of 1 on the left side of the scale, but this should not be considered a reasonable common setting to default to when you aren't sure what rating to select. We have no way of knowing what your customer's application logic is like. To come up with a reasonable sizing estimate, you must work with your customer to select an appropriate rating.

Below are DB2 processing profiles of two applications we use in lab performance studies.

Application A (DayTrader 2.0) consists of moderate DB2 processing and is rated at 3 on a scale of 1 to 10. It consists of:

- 4 to 8 simple SQL calls
- Mostly cursor selects
- On average, 1 update/insert/delete per 10 cursor selects

	# per Tran	Input Columns	Output Columns
Prepare	1.4		
Select	0	0	0
Insert	0.1	9.7	
Update searched	0.1	9.5	
Update cursor	0	0	
Delete searched	0	1	
Delete cursor	0	0	
Open	2.1	0.9	
Fetch	6.8		12.1

Ratio of Read-Only Commits to Update Commits:	7.1
% Hits in Dynamic Statement Cache:	99.9%

Application B consumes 3.5 times more CPU per transaction than Application A, and is rated at 9 on the scale. It consists of:

- 11 to 14 simple SQL calls
- Relatively update intensive
- On average, 1 update/insert/delete per 3.5 cursor selects
- Uses XA global transaction and default JMS messaging

	# per Tran	Input Columns	Output Columns
Prepare	1.8		
Select	0	0	0
Insert	0.3	10.1	
Update searched	1.8	10.1	
Update cursor	0	0	
Delete searched	0.3	1	
Delete cursor	0	0	
Open	4.9	1.6	
Fetch	14.7		18.3

Ratio of Read-Only Commits to Update Commits:	1.0
% Hits in Dynamic Statement Cache:	99.9%

Using these profiles as a general guide, rate the amount and complexity of your DB2 SQL statement processing on a scale of 1 to 10. Do not include here DB2 processing by CICS or IMS transactions.

This tool is intended to estimate the capacity requirements only for WAS/DB2 applications, not for other kinds of DB2 applications.

Push Buttons

Click the **Return** button to return to the primary ***WAS Application Definition*** window with any rating you select by moving the slider with your cursor.

Click the **Cancel** button to return to the primary ***WAS Application Definition*** window without saving the change in the rating. If you use this button instead of the **Return** button, the output windows will not reflect the DB2 rating the slider is left on.

Default

No default, select a rating on the scale of 1 to 10.

Considerations

Support for local DB2 SQL processing is based on WAS and DB2 both running in a single system z/OS image, using Type 2 JDBC/SQLJ local access. This access to DB2 is the most efficient link between WAS and DB2. Select activity #9 on the ***WAS Application Definition*** window and include JDBC / SQLJ Access to Local DB2 to account for the CPU cost to access DB2 locally since Local DB2 SQL Processing does not include the CPU cost to access DB2 with the JDBC / SQLJ Type 2 connector. If the WAS application is accessing DB2 in a different system image using the new Type 4 Universal JDBC / SQLJ connector or DB2 Connect (DRDA), then you should **not** select Local DB2 SQL Processing on the ***WAS Application Definition*** window. Instead, select activity

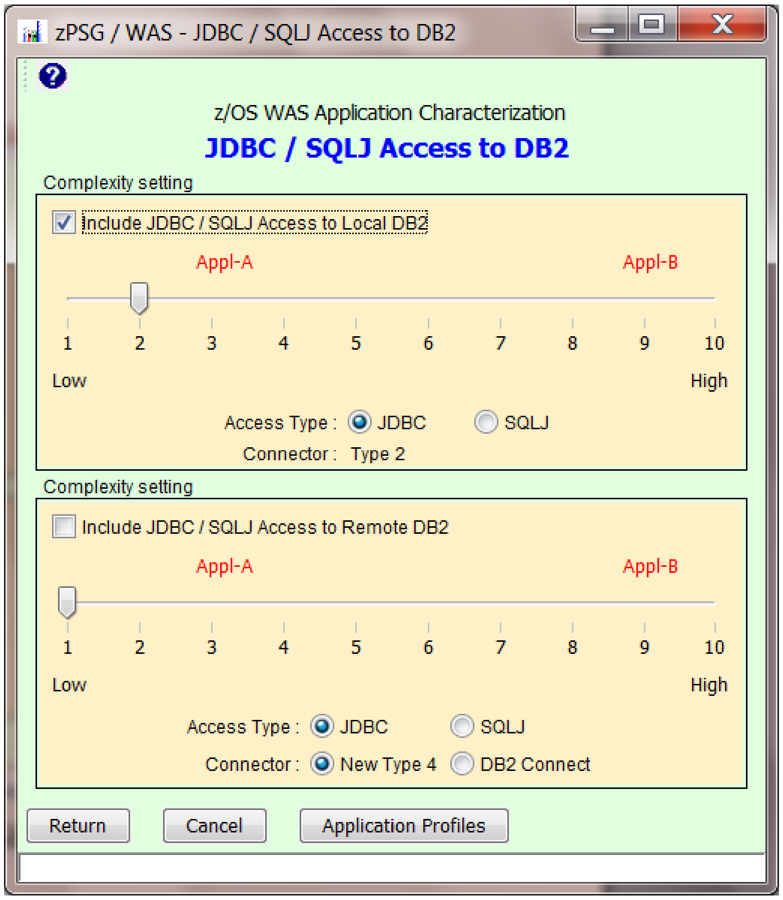
#9 on the ***WAS Application Definition*** window and include JDBC / SQLJ Access to Remote DB2.

The scale represents the complexity of your SQL logic and the amount of data that DB2 must process. Here are some things that might cause you to select a higher or lower application profile rating:

- SQL is a programming language in itself. Calculations and business logic are sometimes performed with just SQL. If your SQL statements are lengthy, contain much logic, or cause DB2 to pass the data multiple times, consider giving your JDBC/SQLJ application profile a higher rating.
- If what you consider a transaction actually consists of more SQL statements than our samples above, you should give your application profile a higher rating.
- If what you consider a transaction consists of a single request to DB2 or a small number of DB2 requests, you should give your application profile a lower rating.
- DB2 stored procedures can contain both SQL and business logic. If your WAS transactions are invoking stored procedures that contain many SQL statements, have much business logic, or are written in Java, you should give your application profile a higher rating, and consider pursuing a separate DB2 sizing for this part of the application.

The two model applications, Application A and Application B, represent two points of reference on the complexity scale. The above information for these applications should give you a feel for how complex they are relative to your application. This should help you select the right place for your application on the scale.

<u>JDBC / SQLJ Access to DB2</u>



z/OS WAS Application Characterization

JDBC / SQLJ Access to DB2

Complexity setting

[✓] Include JDBC / SQLJ Access to Local DB2

Appl-A Appl-B

1 [✓ 2] 3 4 5 6 7 8 9 10

Low High

Access Type : [✓ JDBC] SQLJ

Connector : Type 2

Complexity setting

[ ] Include JDBC / SQLJ Access to Remote DB2

Appl-A Appl-B

[✓ 1] 2 3 4 5 6 7 8 9 10

Low High

Access Type : [✓ JDBC] SQLJ

Connector : [✓ New Type 4] DB2 Connect

This window is displayed when the <u>**Customize**</u> button for <u>JDBC / SQLJ Access to DB2</u> is clicked on the ***WAS Application Definition*** window.

This window is used to define <u>JDBC / SQLJ access to Local DB2</u> for a WAS application running in the same system image as DB2 and/or <u>JDBC / SQLJ Access to Remote DB2</u> for a WAS application running in a different system image than DB2. Access to Local DB2 is only available to a WAS sizing on z/OS. Only the CPU costs associated with JDBC / SQLJ access to DB2 are included in the sizing estimate for this activity. The access type can be defined as JDBC or SQLJ and for remote access to DB2 the connector can be defined as the new Type 4 Universal connector or DB2 Connect (DRDA). For local access to DB2 on z/OS the Type 2 connector is assumed.

Although DB2 SQL statement processing is not included in the CPU capacity requirement for JDBC / SQLJ processing in the WAS system image, the CPU capacity requirement depends on the number and type of SQL statements (even though SQL statement processing costs are not included), the number of rows returned by DB2 to the WAS application, and the complexity of any stored procedures. For this reason you must select a rating on a linear scale of 1 to 10 to represent the amount and complexity of JDBC / SQLJ processing for a typical WAS transaction, based on 2 application profiles for DB2 access. The profiles can be viewed by pressing the **Application Profiles** button.

Description of Input Fields

Toolbar

? button Click this button to go to Help for this window.

Include JDBC / SQLJ Access to Local DB2 **For z/OS only**
(checkbox) Check this box to include local DB2 access.

Include JDBC / SQLJ Access to Remote DB2
(checkbox) Check this box to include remote DB2 access.

Application Complexity Setting

Scale of 1 to 10 with Slider

Move the slider by clicking on it with your cursor and dragging it to the rating you desire. The slider is initially set at a rating of 1 on the left side of the scale, but this should not be considered a reasonable common setting to default to when you aren’t sure what rating to select. We have no way of knowing what your customer’s application logic is like. To come up with a reasonable sizing estimate, you must work with your customer to select an appropriate rating.

Below the rating scales there is a button named **Application Profiles** which when pressed will provide information about the DB2 processing profiles of two applications we use in lab performance studies.

Application A (DayTrader 2.0) consists of moderate DB2 processing and is rated at 3 on a scale of 1 to 10. It consists of:

- 4 to 8 simple SQL calls
- Mostly cursor selects
- On average, 1 update/insert/delete per 10 cursor selects

	# per Tran	Input Columns	Output Columns
Prepare	1.4		
Select	0	0	0
Insert	0.1	9.7	
Update searched	0.1	9.5	
Update cursor	0	0	
Delete searched	0	1	
Delete cursor	0	0	

Open	2.1	0.9	
Fetch	6.8		12.1

Ratio of Read-Only Commits to Update Commits: 7.1
% Hits in Dynamic Statement Cache: 99.9%

Application B consumes 3.5 times more CPU per transaction than Application A, and is rated at 9 on the scale. It consists of:

- 11 to 14 simple SQL calls
- Relatively update intensive
- On average, 1 update/insert/delete per 3.5 cursor selects
- Uses XA global transaction and default JMS messaging

	# per Tran	Input Columns	Output Columns
Prepare	1.8		
Select	0	0	0
Insert	0.3	10.1	
Update searched	1.8	10.1	
Update cursor	0	0	
Delete searched	0.3	1	
Delete cursor	0	0	
Open	4.9	1.6	
Fetch	14.7		18.3

Ratio of Read-Only Commits to Update Commits: 1.0
% Hits in Dynamic Statement Cache: 99.9%

Using these profiles as a general guide, rate the amount and complexity of your DB2 connector processing on a scale of 1 to 10. Do not include here DB2 access by CICS or IMS transactions. This activity is intended to estimate the capacity requirements only for WAS applications accessing DB2 directly via JDBC/SQLJ.

Access type

Select JDBC or SQLJ using the appropriate radio button.

Connector type **Active for remote access only**

Select Type 4 or DB2 Connect using the appropriate radio button.

Note: For local DB2 access, Type 2 connector is always assumed.

Click the appropriate radio button to select the Connector Type for Remote access.

Push Buttons

Click the **Return** button to return to the primary ***WAS Application Definition*** window with the settings you have selected.

Click the **Cancel** button to return to the primary ***WAS Application Definition*** window without saving the current inputs. If you use this button instead of the **Return** button, the output windows will not reflect the settings you have selected.

Click the **Application Profiles** button to see the workload profiles for Appl-A and Appl-B which are on the complexity scale.

Defaults

Complexity setting: No default, select a rating on the scale of 1 to 10.

Access type: JDBC

Connector: Type 2 for local access, Type 4 for remote access

Considerations

This activity is based on WAS running under either z/OS or Linux, using the Type 4 JDBC/SQLJ connector to access to a DB2 system running in a separate image, or for Linux (only), using DB2 Connect to access DB2. DB2 Connect adds about 5% additional CPU cost compared to the Type 4 connector.

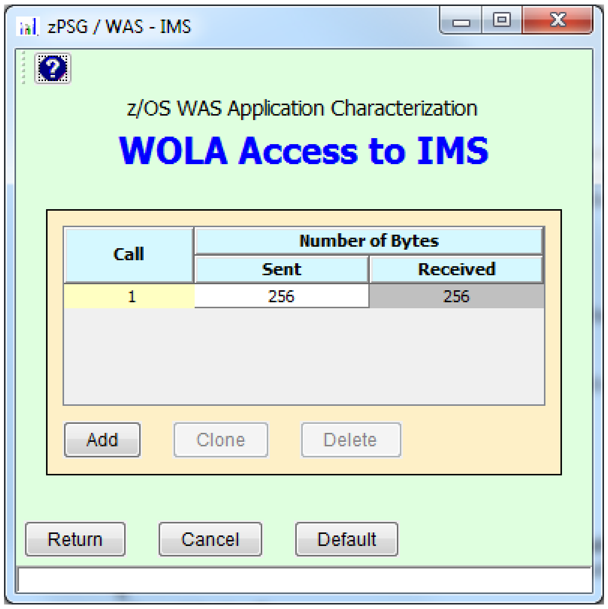
On average, using SQLJ saves about 10% of the CPU cost associated with JDBC access.

The scales represents the complexity of your JDBC/SQLJ processing, which depends on the type of SQL statements involved and the amount of data being returned from DB2. Here are some things that might cause you to select a higher or lower application profile rating:

- SQL is a programming language in itself. Calculations and business logic are sometimes performed with just SQL. If your SQL statements are lengthy, contain much logic, or cause DB2 to pass the data multiple times, consider giving your JDBC/SQLJ application profile a higher rating.
- If what you consider a transaction actually consists of more SQL statements than our samples above, you should give your application profile a higher rating.
- If what you consider a transaction consists of a single request to DB2 or a small number of DB2 requests, you should give your application profile a lower rating.

The two model applications, Application A and Application B, represent two points of reference on the complexity scale. The above information for these applications should give you a feel for how complex they are relative to your application. This should help you select the right place for your application on the scale.

WOLA Access to IMS



Call	Number of Bytes	
	Sent	Received
1	256	256

This window is displayed when the **Customize** button for WOLA Access to IMS is clicked on the ***WAS Application Definition*** window.

JCA (J2EE) connector support for access to IMS is provided by the Optimized Local Adapters (WOLA). Data to be sent to IMS is generally stored in a COMMAREA, which is sent to IMS by the adapter. After the IMS transaction executes, the results are returned to WAS. The amount of data sent to and returned from IMS affects the CPU cost of connector processing.

Description of Input Fields

Toolbar

? button Click this button to go to Help for this window.

Table

Calls column

Not a direct input field. An entry for 1 message is provided. If there are additional messages in a typical transaction, use the **Add** or **Clone** buttons to add entries to the table.

Number of Bytes Sent/Received columns
Enter the number of bytes sent in a COMMAREA from WAS to IMS. The number of bytes received back from IMS to WAS is assumed to be the same number as sent. If there are multiple calls per transaction, use separate rows to describe the different data connector transmissions.

Buttons
Add button
Use this button to add another row to the table with the default values.

Clone button
Highlight a row in the table and use this button to add another row with the same values.

Delete button
Use this button to delete a row from the table.

Return button
Use this button to return to the primary ***WAS Application Definition*** *window*, using any values you have entered or changes you have made.

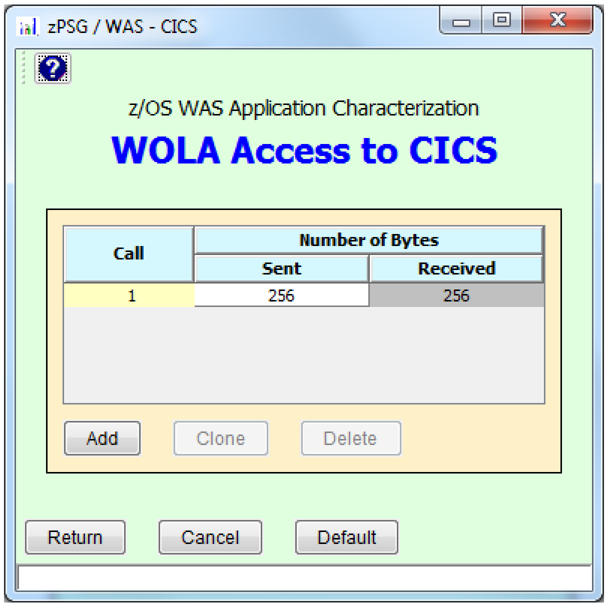
Cancel button
Use this button to return to the primary ***WAS Application Definition*** window, without using any values or changes you have entered.

Default button
Use this button to restore the default values for this activity.

Defaults
The default is 1 call per transaction. 256 bytes are sent out to IMS, the same amount is always returned.

Considerations/Explanation/Things to Think About
zPSG data connector sizing support is based on access to IMS in the same LPAR as WAS. The local interface to IMS Connect is assumed. Connector processing to IMS in another z/OS image would involve some additional capacity requirement. In addition, our sizing estimates assume that your application is built using Rational Application Developer (RAD) for WebSphere Software version 8.5 or higher.

WOLA Access to CICS



z/OS WAS Application Characterization

WOLA Access to CICS

Call	Number of Bytes	
	Sent	Received
1	256	256

This window is displayed when the **Customize** button for WOLA Access to CICS is clicked on the ***WAS Application Definition*** window.

JCA (J2EE) connector support for access to CICS is provided by the WebSphere Optimized Local Adapters (WOLA). Data to be sent to CICS is generally stored in a COMMAREA, which is sent to CICS by the connector. After the CICS transaction executes, the results are returned to WAS. The amount of data sent to and returned from CICS affects the CPU cost of connector processing.

Description of Input Fields

Toolbar

? button Click this button to go to Help for this window.

Table

Calls column

Not a direct input field. An entry for 1 message is provided. If there are additional messages in a typical transaction, use the **Add** or **Clone** buttons to add entries to the table.

Number of Bytes Sent/Received columns
Enter the number of bytes sent in a COMMAREA from WAS to CICS. The number of bytes received back from CICS to WAS is assumed to be the same number as sent. If there are multiple calls per transaction, use separate rows to describe the different data connector transmissions.

Buttons

Add button
Use this button to add another row to the table with the default values.

Clone button
Highlight a row in the table and use this button to add another row with the same values.

Delete button
Use this button to delete a row from the table.

Return button
Use this button to return to the primary ***WAS Application Definition*** *window*, using any values you have entered or changes you have made.

Cancel button
Use this button to return to the primary ***WAS Application Definition*** window, without using any values or changes you have entered.

Default button
Use this button to restore the default values for this activity.

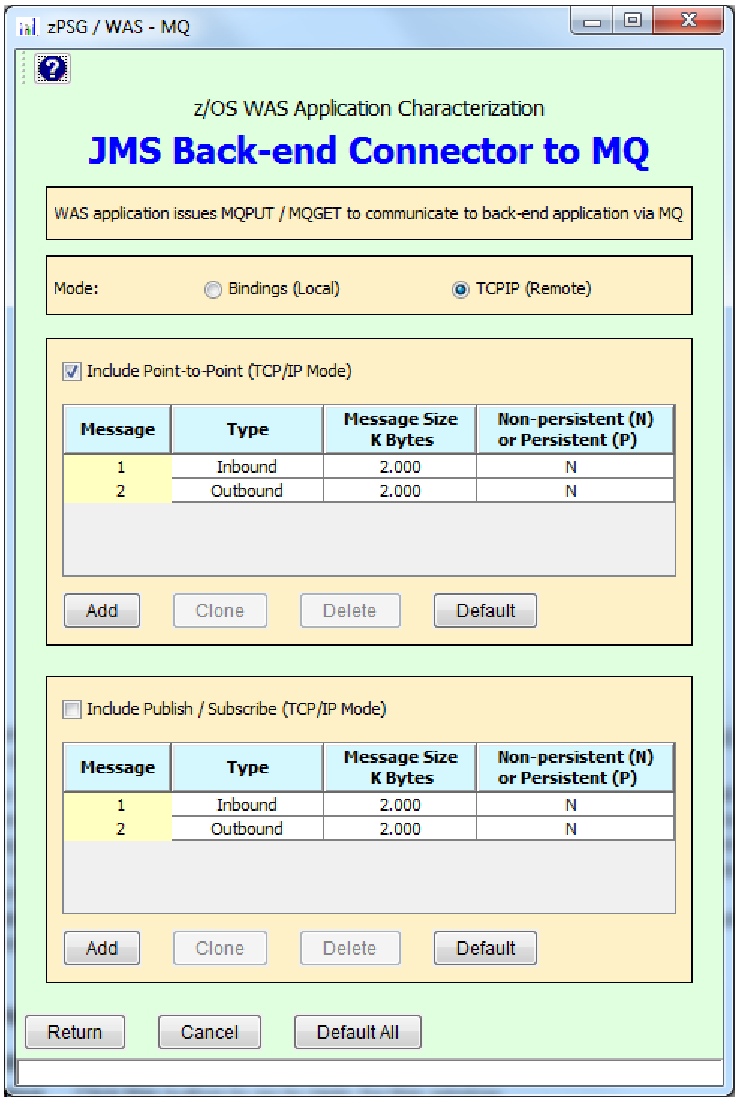
Defaults

The default is 1 call per transaction. 256 bytes are sent out to CICS, is the same amount is always returned. Access to CICS is in Local mode.

Considerations/Explanation/Things to Think About

zPSG sizing support is based on access to CICS in the same LPAR as WAS. Connector processing to CICS in another z/OS image would involve some additional capacity requirement. When CICS is running in same LPAR as WAS, local mode is assumed. Note that CICS running in same z/OS image as WAS, and local mode, are required for 2 phase commit transactions. In addition, our sizing estimates assume that your application is built using Rational Application Developer (RAD) for WebSphere Software version 8.5 or higher.

JMS Back-end Connector to MQ



z/OS WAS Application Characterization

JMS Back-end Connector to MQ

WAS application issues MQPUT / MQGET to communicate to back-end application via MQ

Mode: Bindings (Local) [✓ TCPIP (Remote)]

[✓] Include Point-to-Point (TCP/IP Mode)

Message	Type	Message Size K Bytes	Non-persistent (N) or Persistent (P)
1	Inbound	2.000	N
2	Outbound	2.000	N

[ ] Include Publish / Subscribe (TCP/IP Mode)

Message	Type	Message Size K Bytes	Non-persistent (N) or Persistent (P)
1	Inbound	2.000	N
2	Outbound	2.000	N

This window is displayed when the **Customize** button for JMS Back-end Connector to MQ is clicked on the ***WAS Application Definition*** window.

There are 2 independent sections on this window, 1 for point-to-point messaging and 1 for publish/subscribe messaging. The input fields for them are identical (but the defaults are not).

Mode selection buttons (z/OS only)
Select which mode is being used. Either TCP/IP (remote) or Bindings (local) mode.

Include checkbox. Use these boxes to independently include point-to-point or publish/subscribe messaging in a typical transaction.

Table

Message column
Not a direct input field. An entry for 1 inbound message and 1 outbound message is provided by default. If there are additional messages in a typical transaction, use the **Add** or **Clone** buttons to add entries to the table or the **Delete** button to eliminate a row.

Message Size in k Bytes column
Enter the message sizing in k.

Non-persistent or Persistent column
Enter a "P" for Persistent or an "N" for Non-Persistent. See the discussion above about assumptions regarding the combination of attributes to make messages recoverable and durable.

Buttons

Add button
Use this button to add another row to the table with the default values.

Clone button
Highlight a row in the table and use this button to add another row with the same values.

Delete button
Use this button to delete a row from the table.

Default button
Use this button to restore the default values for either point-to-point or publish/subscribe messaging.

Return button
Use this button to return to the primary ***WAS Application Definition*** *window*, using any values you have entered or changes you have made.

Cancel button
Use this button to return to the primary ***WAS Application Definition*** window, without using any values or changes you have entered.

Default all button
Use this button to restore the all default values for this activity (both point-to-point and publish/subscribe messaging).

General assumptions used for sizing - z/OS

- JMS client is a WAS EJB
- Each message defined includes the CPU cost of a MQPUT or MQGET processing by the WAS application for either an outbound or an inbound JMS message.
- No new connection for each JMS message.
- MQ queue is not SYSPLEX enabled.
- MQ dual logging is assumed.
- Sizing support for point-to-point messaging is limited to message sizes of 4MB or less; support for publish/subscribe messaging is limited to message sizes of 40k or less.
- For Bindings (Local) mode, the sizing includes the processing of the messages by MQ.
- TCP/IP (Remote) mode is the default

General assumptions used for sizing - Linux

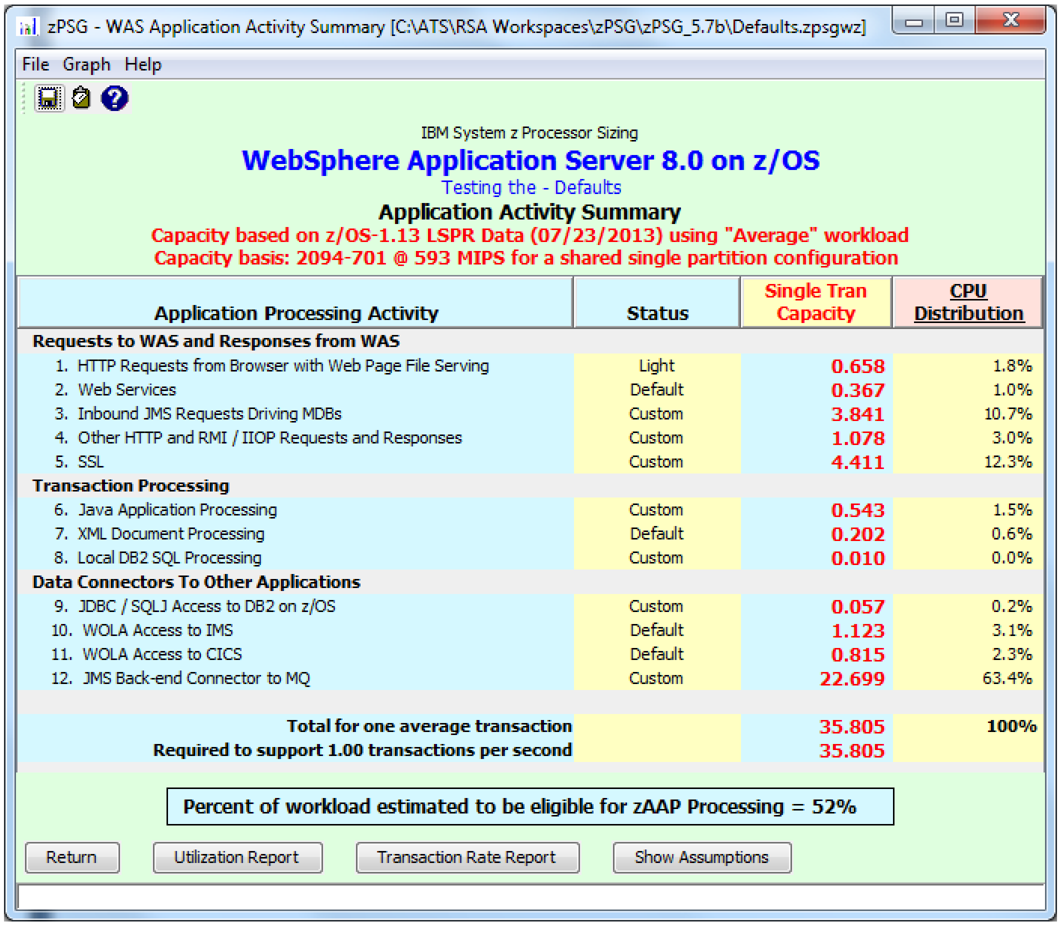
- JMS client is a WAS EJB
- Each message defined includes the CPU cost of a MQPUT or MQGET processing by the WAS application for either an outbound or an inbound JMS message. MQ and back-end application processing are not included.
- Sizing support for point-to-point messaging is limited to message sizes of 4MB or less; support for publish/subscribe messaging is limited to message sizes of 40k or less.
- TCP/IP mode only (Linux JMS messaging does not support Bindings Mode on System z).

Defaults

Default for entire JMS Back-End Connector activity is point-to-point messaging using TCP/IP (remote) mode with a pair of 2k non-persistent, non-transacted (express) messages, 1 outbound and 1 inbound. By default, publish/subscribe messaging is not included.

Default for publish/subscribe messaging, if specifically included, is 1 non-persistent, non-transacted, non durable inbound message of 2k and 1 non-persistent, non-transacted, non-durable outbound message of 2k.

WAS Application Activity Summary


zPSG - WAS Application Activity Summary [C:\ATS\RSA Workspaces\zPSG\zPSG_5.7b\Defaults.zpsgwz]
File Graph Help

IBM System z Processor Sizing
WebSphere Application Server 8.0 on z/OS
Testing the - Defaults
Application Activity Summary
Capacity based on z/OS-1.13 LSPR Data (07/23/2013) using "Average" workload
Capacity basis: 2094-701 @ 593 MIPS for a shared single partition configuration

Application Processing Activity	Status	Single Tran Capacity	CPU Distribution
Requests to WAS and Responses from WAS			
1. HTTP Requests from Browser with Web Page File Serving	Light	0.658	1.8%
2. Web Services	Default	0.367	1.0%
3. Inbound JMS Requests Driving MDBs	Custom	3.841	10.7%
4. Other HTTP and RMI / IIOP Requests and Responses	Custom	1.078	3.0%
5. SSL	Custom	4.411	12.3%
Transaction Processing			
6. Java Application Processing	Custom	0.543	1.5%
7. XML Document Processing	Default	0.202	0.6%
8. Local DB2 SQL Processing	Custom	0.010	0.0%
Data Connectors To Other Applications			
9. JDBC / SQLJ Access to DB2 on z/OS	Custom	0.057	0.2%
10. WOLA Access to IMS	Default	1.123	3.1%
11. WOLA Access to CICS	Default	0.815	2.3%
12. JMS Back-end Connector to MQ	Custom	22.699	63.4%
Total for one average transaction		35.805	100%
Required to support 1.00 transactions per second		35.805	

Percent of workload estimated to be eligible for zAAP Processing = 52%

This window is displayed when the **Summary Report** button is clicked on the primary ***WAS Application Definition*** window. It shows a breakdown of the CPU per transaction for the various kinds of activity included in the typical web transaction.

Menu bar

File

Output	Write contents to a flat (PRN) file.
Copy	Write contents to Window's clipboard

Graph Generates a pie chart showing the distribution of application activity

Help

Context Help (F1)	Help for this window
About zPSG	Product information

Toolbar

1st button
Click this button to send sizing information to a PRN file for processing outside of zPSG.

2nd button
Click this button to send sizing information to the clipboard, so that you can copy it into a note or other document.

? button
Click this button to go to Help for this window.

Application Processing Activity column

Lists the activities that can be included in a typical transaction (from the primary ***WAS Application Definition*** window).

Status column

Reflects which activities are included and, if so, whether the default values are used for each item or whether they have been configured.

Single Tran Capacity column

Reflects the amount of CPU (as represented by the Capacity Rating) for each activity, and at the bottom for the transaction as a whole. The last row in this column shows the capacity rating for the entire workload, i.e. when the capacity rating of the transaction is multiplied by the transaction rate.

CPU Distribution column

Shows the percentage of the CPU/transaction used by each activity.

Percent of workload estimated to be eligible for zAAP Processing = (z/OS only)

Shows the estimated percentage of Java content for the typical web transaction as defined. Percentages of Java content were computed in all the performance lab measurements done to support WAS sizing, including the full-function workloads used to establish the Java application processing scale, and specific workloads used to establish sizing support for functions like web services, JMS, XML document processing, web page file serving, and data connectors. Depending on what you included in your typical transaction, the percentage per transaction will vary. These percentages reflect the amount of CPU that we estimate you can offload to zAAP, assuming sufficient zAAP capacity to handle the load. You can generate an estimate of zAAP capacity requirements using the zAAP Capacity Estimator available from the CP Calculator menu on the ***Product Selection*** window.

Notice Concerning Specialty Engines

Neither **zPSG** nor this document provides descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zAAP. zIIP, and IFL). IBM authorizes customers to use IBM Specialty Engines only to process Eligible Workloads of specific Programs expressly authorized by IBM. These programs are specified in the "Authorized Use Table for IBM Machines", found at:

www.ibm.com/systems/support/machine_warranties/machine_code/aut.html

No other workload processing is authorized for execution on an SE.

IBM offers Specialty Engines at a lower price than General Processors/Central Processors because customers are authorized to use Specialty Engines only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Push Buttons

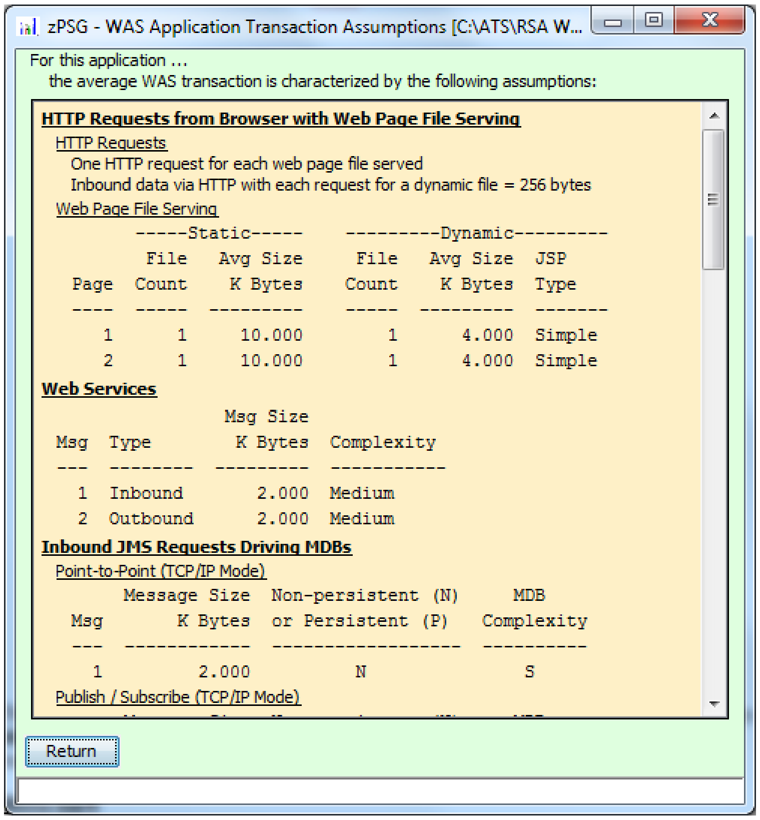
Click the **Return** button to return to the primary ***WAS Application Definition*** input window.

Click the **Utilization Report** button to go to the ***WAS Processor Capacity Projections - Processor Utilization*** output window.

Click the **Transaction Rate Report** button to go to the ***WAS Processor Capacity Projections - Transaction Rate Supported*** output window.

Click the **Show Assumptions** button to see a list of the assumptions for the sizing in the ***WAS Application Transaction Assumptions*** window.

WAS Application Transaction Assumptions



zPSG - WAS Application Transaction Assumptions [C:\ATS\RSA W...

For this application ...
the average WAS transaction is characterized by the following assumptions:

HTTP Requests from Browser with Web Page File Serving

HTTP Requests

One HTTP request for each web page file served
Inbound data via HTTP with each request for a dynamic file = 256 bytes

Web Page File Serving

```
        -----Static-----    ---------Dynamic---------
         File   Avg Size     File   Avg Size  JSP
 Page   Count    K Bytes    Count    K Bytes  Type
 ----   -----  ---------    -----  ---------  -------
    1       1     10.000        1      4.000  Simple
    2       1     10.000        1      4.000  Simple
```

Web Services

```
                 Msg Size
Msg  Type         K Bytes  Complexity
---  --------   ---------  -----------
  1  Inbound        2.000  Medium
  2  Outbound       2.000  Medium
```

Inbound JMS Requests Driving MDBs

Point-to-Point (TCP/IP Mode)

```
       Message Size  Non-persistent (N)     MDB
 Msg        K Bytes  or Persistent (P)   Complexity
 ---   ------------  ------------------  ----------
   1          2.000          N               S
```

Publish / Subscribe (TCP/IP Mode)

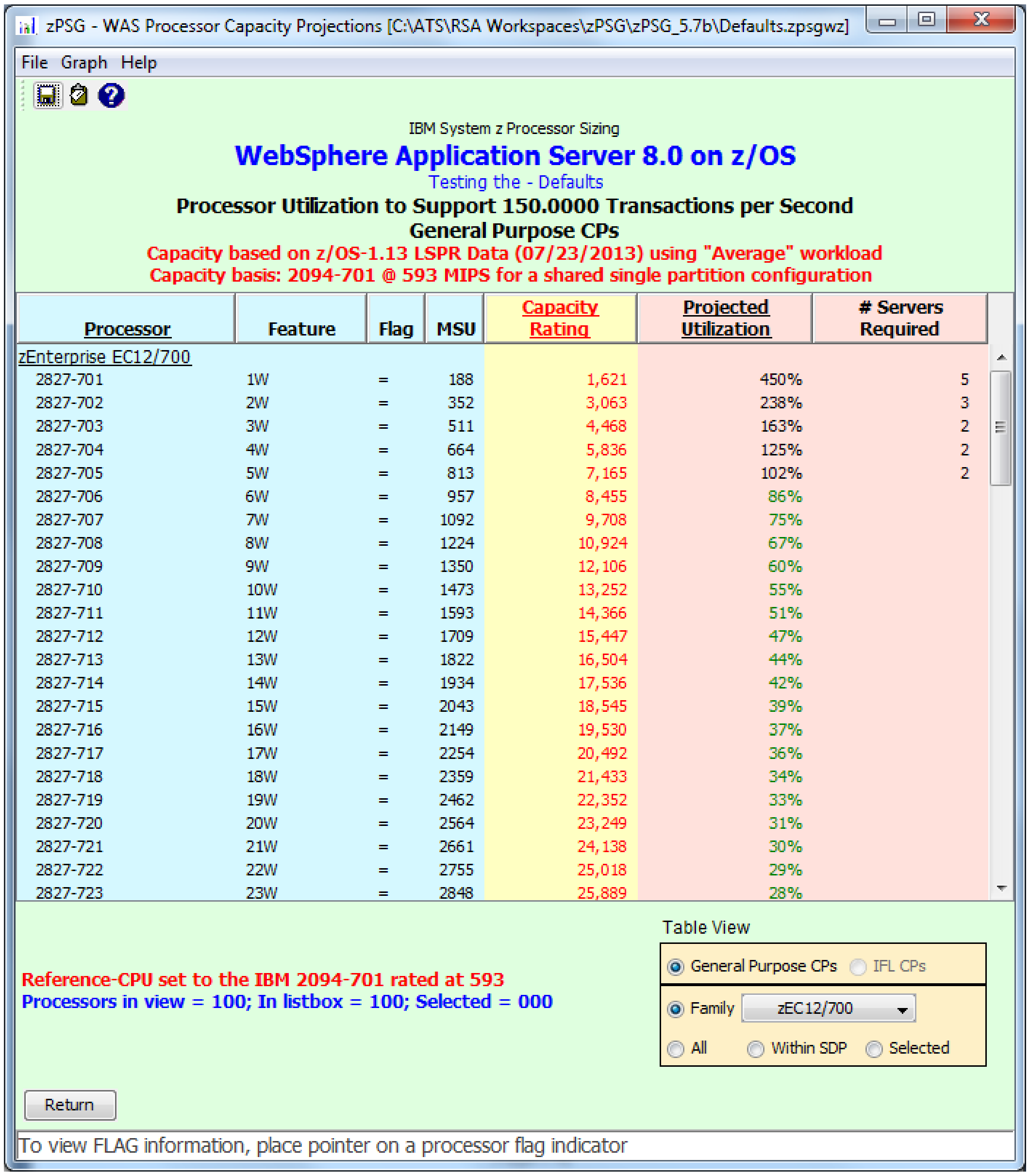
Return

This window is displayed when the **Show Assumptions** button is clicked on the ***WAS Application Activity Summary*** window.

All assumptions as listed will be included when generating output for the ***Summary*** window.

WAS Processor Utilization

WAS Processor Capacity Projections



zPSG - WAS Processor Capacity Projections [C:\ATS\RSA Workspaces\zPSG\zPSG_5.7b\Defaults.zpsgwz]

IBM System z Processor Sizing

WebSphere Application Server 8.0 on z/OS

Testing the - Defaults

Processor Utilization to Support 150.0000 Transactions per Second

General Purpose CPs

Capacity based on z/OS-1.13 LSPR Data (07/23/2013) using "Average" workload

Capacity basis: 2094-701 @ 593 MIPS for a shared single partition configuration

Processor	Feature	Flag	MSU	Capacity Rating	Projected Utilization	# Servers Required
zEnterprise EC12/700						
2827-701	1W	=	188	1,621	450%	5
2827-702	2W	=	352	3,063	238%	3
2827-703	3W	=	511	4,468	163%	2
2827-704	4W	=	664	5,836	125%	2
2827-705	5W	=	813	7,165	102%	2
2827-706	6W	=	957	8,455	86%	
2827-707	7W	=	1092	9,708	75%	
2827-708	8W	=	1224	10,924	67%	
2827-709	9W	=	1350	12,106	60%	
2827-710	10W	=	1473	13,252	55%	
2827-711	11W	=	1593	14,366	51%	
2827-712	12W	=	1709	15,447	47%	
2827-713	13W	=	1822	16,504	44%	
2827-714	14W	=	1934	17,536	42%	
2827-715	15W	=	2043	18,545	39%	
2827-716	16W	=	2149	19,530	37%	
2827-717	17W	=	2254	20,492	36%	
2827-718	18W	=	2359	21,433	34%	
2827-719	19W	=	2462	22,352	33%	
2827-720	20W	=	2564	23,249	31%	
2827-721	21W	=	2661	24,138	30%	
2827-722	22W	=	2755	25,018	29%	
2827-723	23W	=	2848	25,889	28%	

Table View: [✓ General Purpose CPs] IFL CPs

[✓ Family] zEC12/700; All; Within SDP; Selected

Reference-CPU set to the IBM 2094-701 rated at 593

Processors in view = 100; In listbox = 100; Selected = 000

To view FLAG information, place pointer on a processor flag indicator

This window is displayed when the **Utilization** button is clicked on the ***WAS Application Definition*** window or the **Utilization Report** button is clicked on the ***WAS Application Activity Summary*** window.

Menu bar

File

Output	Write report contents to a flat (PRN) file.
Copy	Write report contents to Window's clipboard

Graph (for processors currently selected in table)

Capacity	Generate bar graph depicting capacity values
Utilization	Generate bar graph showing utilization on selected processors

Help

Context Help (F1)	Help for this window
About zPSG	Product information

Toolbar

1st button
Click this button to send sizing information to a PRN file, for processing outside of zPSG.

2nd button
Click this button to send sizing information to the clipboard, so that you can copy it into a note or other document.

? button
Click this button to go to Help for this window.

Table

***Processor* column**
A list of all processor models supported in zPSG

***Feature* column** **For z/OS & Linux**
Using the General Purpose CPs option under Table View, a designation of how many general purpose processing engines (CPs) for this entry. For example, 4W ("W" is short for "way") indicates 4 CPs or engines. Also see ***Flag*** column below.

***Feature* column** **For Linux only**
Using the IFL CPs option under Table View, a designation of how many IFL engines for this entry. For example, 4W IFL ("W" is short for "way") indicates 4 IFL engines. Also see ***Flag*** column below.

***Flag* column**
If you place your cursor on a row in this column, an explanatory message about the System z model designation and the number of CP or IFL engines for the entry.

***MSU* column**
Only for the General Purpose CPs Table View (does not apply to IFLs). Shows the MSU rating assigned to the number of CP engines for this entry.

***Capacity Rating* column**
The capacity ratings reflect the relative capacity of each processor table entry to the reference-CPU and its capacity rating assigned on the Reference-CPU window. When **zPSG** is started the reference-CPU will be set to a 2094-701 (a z9 EC/700 processor with 1 general purpose CP) with a capacity rating of 593 MIPS.

***Projected Utilization* column**
Shows the estimated CPU% for each processor entry in the table, based on the transaction described and the transaction rate(s) provided. This is the primary output for a sizing.

# *Servers Required* column
If the estimated CPU% is greater than 100% (and therefore cannot fit on the processor), this column reflects the number of these models that would be needed to accommodate the load.

Table View Options Box

Click a radio button in each section to customize the processor entries shown in the table:

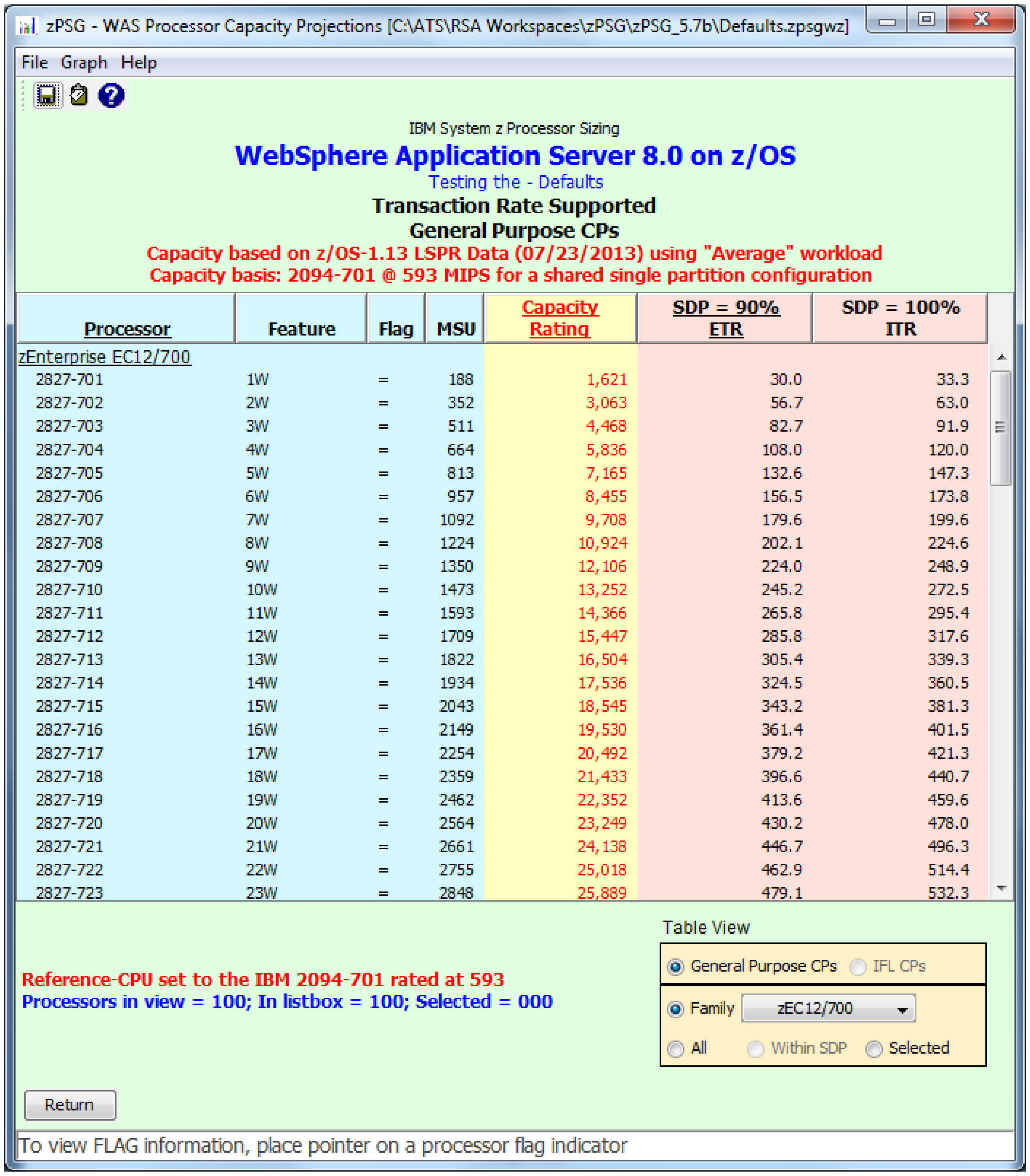
- **General Purpose CPs** shows entries with some number of general purpose CP engines
- **IFL CPs** shows entries with some number of IFL engines (for Linux only)
- **Family** shows all processor models for the family selected (Default)
- **All** shows all processor models supported in zPSG
- **Within SDP** shows all models that can accommodate the load within the Saturation Design Point
- **Selected** shows only selected models. Models are selected by clicking on the entry while holding down the Ctrl key on your keyboard.

Return button

Click this button to return to the primary ***WAS Application Definition*** window.

<u>WAS Transaction Rate Supported</u>

WAS Processor Capacity Projections


zPSG - WAS Processor Capacity Projections [C:\ATS\RSA Workspaces\zPSG\zPSG_5.7b\Defaults.zpsgwz]
File Graph Help

IBM System z Processor Sizing

WebSphere Application Server 8.0 on z/OS

Testing the - Defaults

Transaction Rate Supported

General Purpose CPs

Capacity based on z/OS-1.13 LSPR Data (07/23/2013) using "Average" workload

Capacity basis: 2094-701 @ 593 MIPS for a shared single partition configuration

Processor	Feature	Flag	MSU	Capacity Rating	SDP = 90% ETR	SDP = 100% ITR
zEnterprise EC12/700						
2827-701	1W	=	188	1,621	30.0	33.3
2827-702	2W	=	352	3,063	56.7	63.0
2827-703	3W	=	511	4,468	82.7	91.9
2827-704	4W	=	664	5,836	108.0	120.0
2827-705	5W	=	813	7,165	132.6	147.3
2827-706	6W	=	957	8,455	156.5	173.8
2827-707	7W	=	1092	9,708	179.6	199.6
2827-708	8W	=	1224	10,924	202.1	224.6
2827-709	9W	=	1350	12,106	224.0	248.9
2827-710	10W	=	1473	13,252	245.2	272.5
2827-711	11W	=	1593	14,366	265.8	295.4
2827-712	12W	=	1709	15,447	285.8	317.6
2827-713	13W	=	1822	16,504	305.4	339.3
2827-714	14W	=	1934	17,536	324.5	360.5
2827-715	15W	=	2043	18,545	343.2	381.3
2827-716	16W	=	2149	19,530	361.4	401.5
2827-717	17W	=	2254	20,492	379.2	421.3
2827-718	18W	=	2359	21,433	396.6	440.7
2827-719	19W	=	2462	22,352	413.6	459.6
2827-720	20W	=	2564	23,249	430.2	478.0
2827-721	21W	=	2661	24,138	446.7	496.3
2827-722	22W	=	2755	25,018	462.9	514.4
2827-723	23W	=	2848	25,889	479.1	532.3

Reference-CPU set to the IBM 2094-701 rated at 593
Processors in view = 100; In listbox = 100; Selected = 000

Table View
[✓ General Purpose CPs] IFL CPs
[✓ Family] zEC12/700
All Within SDP Selected

Return

To view FLAG information, place pointer on a processor flag indicator

This window is displayed when the **<u>Transaction Rate</u>** button is clicked on the ***WAS Application Definition*** window or the **<u>Transaction Rate Report</u>** button is clicked on the ***WAS Application Activity Summary*** window.

Menu bar

File

Output	Write report contents to a flat (PRN) file.
Copy	Write report contents to Window's clipboard

Graph (for processors currently selected in table)

Capacity	Generate a bar graph depicting capacity values
ETR	Generate bar graph showing transaction rate supported at SDP
ITR	Generate bar graph showing maximum transaction rate supported

Help

Context Help (F1)	Help for this window
About zPSG	Product information

Toolbar

1st button
Click this button to send sizing information to a PRN file, for processing outside of zPSG.

2nd button
Click this button to send sizing information to the clipboard, so that you can copy it into a note or other document.

? **button**
Click this button to go to Help for this window.

Table

Processor **column**
A list of all processor models supported in zPSG

Feature **column** **For z/OS & Linux**
Using the General Purpose CPs option under Table View, a designation of how many general purpose processing engines (CPs) for this entry. For example, 4W ("W" is short for "way") indicates 4 CPs or engines. Also see **Flag** column below.

Feature **column** **For Linux only**
Using the IFL CPs option under Table View, a designation of how many IFL engines for this entry. For example, 4W IFL ("W" is short for "way") indicates 4 IFL engines. Also see **Flag** column below.

Flag **column**
If you place your cursor on a row in this column, an explanatory message about the System z model designation and the number of CP or IFL engines for the entry.

***MSU* column**
Only for the General Purpose CPs Table View (does not apply to IFLs). Shows the MSU rating assigned to the number of CP engines for this entry.

***Capacity Rating* column**
The capacity ratings reflect the relative capacity of each processor table entry to the reference-CPU and its capacity rating assigned on the Reference-CPU window. When **zPSG** is started the reference-CPU will be set to a 2094-701 (a z9 EC/700 processor with 1 general purpose CP) with a capacity rating of 593 MIPS.

SDP= xx % -- ETR column
Shows the transaction rate for the application that can be supported within the Saturation Design Point specified on the primary ***WAS Application Definition*** window (the default SPD is 90%). ETR stands for External Throughput Rate, which is a standard System z term for transaction rate.

SDP=100% -- ITR column
Shows the transaction rate for the application that can be supported at 100% CPU. ITR stands for Internal Throughput Rate, which is a standard System z term indicating the throughput that can be achieved at 100% CPU. ITR is computed by dividing the ETR by the CPU% (expressed as a decimal). This is the way to correctly rate the processor capacity of each entry in the processor table for this workload (as opposed to MIPS ratings, which are generally erroneous).

Table View Options Box

Click a radio button in each section to customize the processor entries shown in the table:

- **General Purpose CPs** shows entries with some number of general purpose CP engines
- **IFL CPs** shows entries with some number of IFL engines (for Linux only)
- **Family** shows all processor models for the family selected (Default)
- **All** shows all processor models supported in zPSG
- **Within SDP** shows all models that can accommodate the load within the Saturation Design Point
- **Selected** shows only selected models. Models are selected by clicking on the entry while holding down the Ctrl key on your keyboard.

Return button

Click this button to return to the primary ***WAS Application Definition*** window.

WAS Sizing Assistance

Here are instructions for accessing the System z questionnaire and submitting WAS sizing requests to Techline. Note that on the Techline websites there are sizing questionnaires for distributed platforms in addition to System z questionnaires. Be sure to use System z questionnaires for System z sizing requests. The questions and sizing methodologies are different from distributed platforms.

For IBMers:

1. Obtain the latest copy of the WAS sizing questionnaire for System z from the following website:
   - http://w3-03.ibm.com/support/techline/sizing/swsz.html
2. Submit a sizing request to Techline using the instructions found in the sizing questionnaire.

For Business Partners:

1. Obtain the latest copy of the WAS sizing questionnaire for System z via:
   - Phone: Call PartnerLine at 1-800-426-9990 (US and Canada)
   - Email: pwcs@us.ibm.com
   - Online: http://www.ibm.com/partnerworld/techline
2. Submit a sizing request to Techline using the instructions found in the sizing questionnaire.

WAS Glossary of Terms

Bindings Mode Connection
When a JMS connection is made in bindings mode, MQ JMS uses the Java Native Interface (JNI) to call the MQ Queue Manager directly rather than communicating over TCP/IP. This connection mode is much more efficient when the sender and receiver reside in the same image of z/OS. Connections that require TCP/IP are called TCP mode connections.

BMP
A type of entity bean with Bean Managed Persistence. This means that the programmer must add code to persist the contents of the entity bean to the data base.

Cached Handshake
See SSL Handshake.

CCF
Crypto Co-Processor Facility. On S/390 and z900 processor models, 1 or 2 CCFs are included on every processor. They can be used to off-load some SSL processing from the general CPs. Processing can be off-loaded for full SSL handshakes, which reduces that CPU cost by 90% or more, and for TDES encryption & decryption, which reduces that cost by about 50%. CCFs are supported by SSL under z/OS but not under Linux. See Crypto Hardware.

Cipher
See SSL.

Client Authentication
See SSL Client Authentication.

CMP
A type of entity bean with Container Managed Persistence. Using this type of entity bean, the EJB container is responsible for persisting bean contents to the data base.

Crypto Hardware
Either co-processors (CCFs or CFAs) or cards (PCICA, PCICC, PCIXCC) installed in zSeries processors that off-load some SSL processing from the general CP engines.

Cursor
See DB2 Cursor

Data connector
Software that provides support for communication between WAS and back-end applications. Data connectors are used to send transactions or requests, with the accompanying input data and parameters, to a back-end application like CICS or IMS, and to return the transaction response or request results to WAS.

DB2 Connect
The IBM middleware product that provides access from WAS to DB2 data bases running in separate system images from WAS when ASCII to EBCDIC translation is needed.

DB2 Cursor
An API used when multiple rows (records) may be returned by DB2 for a SQL select (read) statement. The API consists of a Declare Cursor, an Open Cursor which initiates the building of the result set of rows by DB2, and a processing loop of Fetch to return each row to the application. Cursors may be open for read only or for update.

DOM
Document Object Model. When you parse an XML document using DOM, you create a tree structure (a program object) in memory, representing the contents of the XML document. The programmer can navigate the tree structure and add, modify, or delete its elements. DOM parsing uses more CPU and more memory than SAX parsing.

DTD
Document Type Definition. Used in XML validation processing. A DTD describes the grammar that constrains an XML document. If, for example, an XML-format personnel file contains entries for many employees, each of which must have 1 social security number, the DTD would contain a rule enforcing the occurrence of 1, and only 1, SSN per employee. The rules that may be described using a DTD are fairly limited in scope. For more extensive control over the contents of an XML document, use an XML Schema instead of a DTD.

EJB
Enterprise Java Bean. This is a specialized Java bean which is architected to provide enterprise-class behavior (transactional support, security, etc.). EJB support is one of the technologies in the J2EE specification.

EJB Container
The EJB Container in WAS provides the runtime environment for enterprise beans.

Entity Bean
A type of EJB which represents permanent data. An entity bean persists its contents to the data base.

Express Message
Also called non-persistent message. Guaranteed to be delivered by MQ at most once, unless there is a system failure. Not hardened to DASD. Deleted when receipt is acknowledged.

Full Handshake (aka non-cached handshake**)**
See SSL Handshake.

Handshake
See SSL Handshake.

Hashing Algorithm
See SSL.

HTTP, HTTPS
HyperText Transfer Protocol, HyperText Transfer Protocol Secure. HTTP is the protocol used for non-SSL communications on the web. HTTPS is for SSL communications.

IMS Connect
An IBM product which provides TCP/IP access to IMS. Recent versions of IMS Connect also provide local mode access to IMS applications on the same system as WAS.

IMS Connector for Java
Data connector runtime support for accessing IMS transactions from WAS applications using J2C (JCA) connector technology.

Java Class
A definition for a certain type of Java object.

Java Method
One instance of a Java class or object.

Java Object
One instance of a Java class. For example, if I have a class called “Animal”, I might create an instance of “Animal” called “Rover”, to represent my dog.

JCA Connector
A means for a WAS application to interact with other system components (CICS, IMS, MQ). A JCA connector conforms to the Java Connector Architecture.

JDBC
Java Data Base Connectivity. JDBC is commonly used to access data in DB2, or other relational data bases, from Java applications.

JMS
Java Message Service. A peer to peer communication facility that can be used by software components or applications, usually in conjunction with MQ Series.

JSP
Java Server Page. JSPs are similar to static HTML pages, but they provide a programming interface which can be used to add dynamic content to the page.

J2EE
Java 2 Platform, Enterprise Edition. The server side platform which provides standard support for EJBs and other enterprise-class technologies in Java.

Local Mode
In the context of JCA Connectors, local mode refers to a means of accessing CICS or IMS without using TCP/IP sockets. Local mode is generally more efficient since it is optimized to exploit the fact that the caller and callee are on the same system.

MQ Message
A string of bytes that is meaningful to the applications that use it

MQ Queue
A named data structure for holding messages until they are retrieved by an application. Multiple senders and receivers can be associated with a single queue.

MQ Queue Manager
A named group of address spaces that run as a z/OS subsystem and manage the resources associated with WebSphere MQ. Applications connect to a Queue Manager using its name.

Parse, parser, parsing
A parser is a program that facilitates the interpretation of XML documents, and the extraction of XML data.

PCICA Card
Peripheral Component Interconnect (PCI) Cryptographic Accelerator card. Offloads some SSL handshake processing from general CP engines on zSeries.

PCICC Card
Peripheral Component Interconnect (PCI) Cryptographic Coprocessor card. Offloads some SSL processing from general CP engines on zSeries.

PCIXCC Card
Peripheral Component Interconnect Extended (PCIX) Cryptographic Coprocessor card. Offloads some SSL processing from general CP engines on zSeries.

Persistent Message
A persistent message is guaranteed to be delivered by MQ once and only once. It must be written to a file or a database to guarantee delivery.

Point-To-Point Messaging
This messaging model enables the delivery of an MQ message to only one recipient, also called a consumer.

Publish/Subscribe Messaging
This messaging model supports the delivery of an MQ message to multiple recipients called topic subscribers.

Queue Manager
See MQ Queue Manager.

RMI/IIOP
Remote Method Invocation using CORBA's communication protocol, IIOP. IIOP stands for Internet InterORB Protocol. Requests to WAS coming from Java clients and other WASs can use RMI/IIOP, which uses less CPU than HTTP requests.

SAX
Simple API for XML. A type of XML parsing. SAX parsing makes the contents of the XML document available to the application through a series of callbacks which occur as the parser scans and interprets the document. For example, the parser gives the application control when it encounters a "start element tag", so that subsequent processing decisions can be based on the tag elements. Callback processing is defined by user supplied handlers which are registered with the parser. During SAX parsing, the XML document is processed sequentially. Unlike DOM, SAX does not allow the program to revisit already parsed message segments unless they have been explicitly saved by application code. It is not possible to modify the original XML document. SAX parsing uses less CPU and less memory than DOM parsing.

Schema
Used in XML validation processing. A schema is used to describe the grammar that constrains an XML document. If, for example, an XML-format personnel file contains entries for many employees, each of which must have 1 social security number specified as ### - ## - ####, the Schema would contain a rule enforcing the occurrence of 1, and only 1, SSN per employee in the prescribed format. XML Schemas provide the ability to exercise a high degree of control over the contents of an XML document. Validation using a Schema does, however, generally require more CPU than validation using a DTD.

Servlet
Java code which can be run in WAS (on the server) in response to an HTTP request.

Session Bean
A type of EJB which represents work to be done on behalf of a particular caller. Session beans can be stateful (saving information from call to call) or stateless (saving no status from call to call).

SOAP
Simple Object Access Protocol.
1. SOAP is a W3C specification which provides a standard for using XML to exchange structured and typed information between peers in a decentralized, distributed environment.
2. SOAP is also the name of the WebSphere Web Services implementation supported in WAS 4.0 and WAS 5.0. The new Web Services support provided by WAS 5.0.2 performs significantly better (uses less CPU) than the original SOAP support. (and it also conforms to SOAP specifications).

SQLJ
Standard Query Language for Java. Another means (in addition to JDBC) to access data in DB2, or other relational data bases, from Java applications. In general, SQLJ access uses less CPU than JDBC, but cannot be dynamically created.

State, Stateful, Stateless
Many client interactions cannot be completed with a single request, requiring several requests to complete. For these multi-request interactions, it’s often necessary to retain client and status information from request to request. This retained information is often referred to as “state”. Session beans which retain state from request to request are called stateful session beans. Session beans which do not retain state are called stateless session beans, and they tend to consume less CPU than statefull session beans.

TCP Mode (or Client Mode) Connection
When a JMS connection is made in TCP mode, JMS uses TCP/IP to call the MQ Queue Manager rather than communicating over the Java Native Interface as it does in bindings mode. With TCP Mode Connections, the MQ Queue Manager does not have to be on the same server, or indeed the same platform.

Transacted Session
This option is used to group a series of messages into an atomic unit of work. All messages in the work unit either succeed or fail. The application server commits the session. If the application server detects an error, it may roll back the transaction. The message is not actually sent until the transaction is committed. The next transaction begins after a call to either commit or rollback.

TripleDES/SHA
The SSL cipher and hashing algorithm that provides the highest level of security, generally used by financial institutions and government agencies with high security requirements. TDES was developed by IBM and both the full handshake and encryption/decryption processing are supported by crypto hardware. Although some processing is offloaded from the general CP engines, TDES/SHA still uses significantly more CPU than RC4/MD5.

Validation
See XML Validation

W3C
World Wide Web Consortium (w3c.org). The W3C is responsible for the creation and advancement of standard web-based technologies.

Web Container
The web container in WAS handles requests for servlets, JSPs, and other files that include server-side code.

Web Services
Web Services is the name given to communication that employs the SOAP standard for messaging. SOAP messages are XML documents containing certain required elements. They enable potential users of applications to find and invoke applications without the need to understand their implementation and underlying structure. Web services uses SAX parsing.

Web Transaction
This is a term we use to refer to any sequence of WAS activity that is repeatable and you want to use as the unit of work for projecting capacity requirements. In most cases, it is based on a business transaction. A web transaction can involve multiple interactions with WAS, any number of Java servlets/EJBs, multiple access to DB2, and multiple data connectors to back-end applications like CICS or IMS. Nothing inherent in WAS dictates what the scope of a transaction is. The important thing is to match your transaction rate with the scope of a web transaction that you choose., i.e. if your web transaction is long and involves a number of activities, the transaction rate would be lower than if you break up this sequence of activity into shorter web transactions.

XML
Extended Markup Language.

XML Attribute
A subcomponent of an XML element. Attributes are specified within element start tags or empty element tags. In the following example, **productId**
is an attribute:

<productName **productId="123abc"**>Whistler Tea Kettle</productName>

XML Element
A subcomponent of an XML document. The example below represents an element called **productName**:

<productName productId="123abc">Whistler Tea Kettle</productName>

XML Validation
Validation is the process used by an XML parser to insure that the contents of an XML document conform to the rules in an associated DTD or Schema.

XSL Transformation

A type of XML processing used to create an XML document.