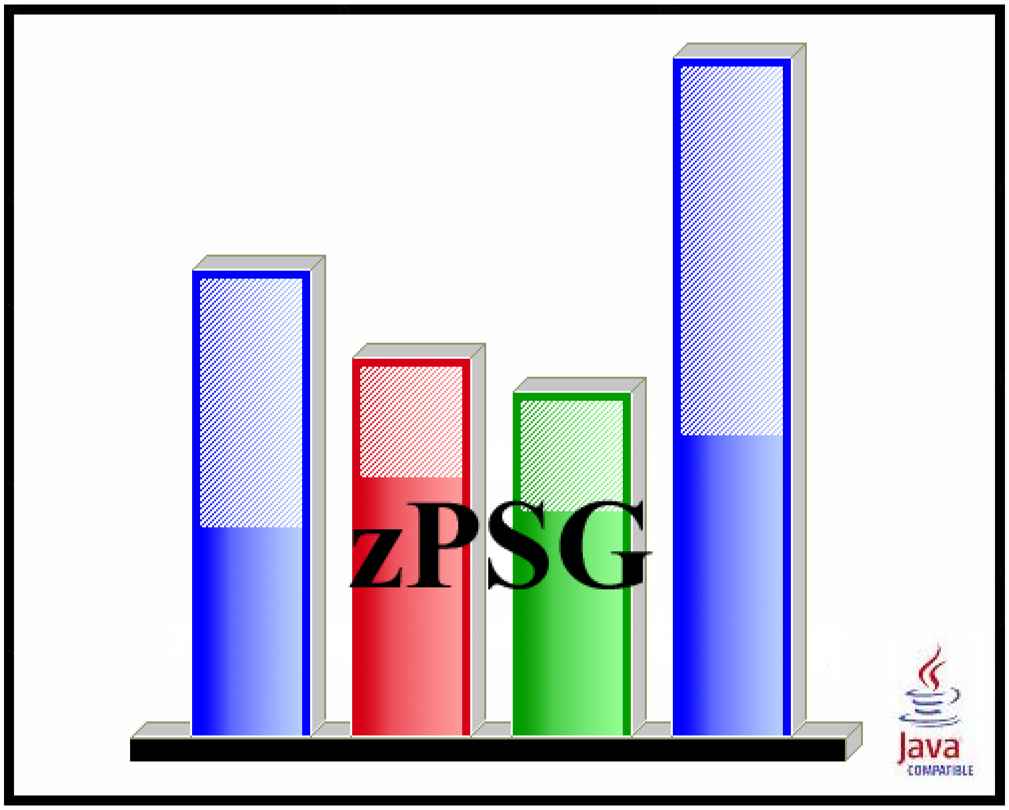
Processor Selection Guide

for

IBM System z

zPSG User's Guide for WebSphere Message Broker



Version 5.4

zPSG WMB UG V54 2013d03.doc

April 25, 2013

WebSphere Message Broker

z/OS and Linux

This tool provides estimates of System z processor capacity for applications using WebSphere Message Broker (WMB) V8.0 on z/OS or Linux for System z.

See the WMB Glossary of Terms for a definition of terms used in describing WMB applications.

WebSphere Message Broker belongs to a family of business integration products that is available from IBM.

Business integration is the coordination and cooperation of all your business processes and applications. It involves bringing together the data and process intelligence in your enterprise, and harnessing these so that your applications and your users can achieve their business goals.
Business integration means that:

- You can connect customers, suppliers, partners, and service providers, with continuing security and control, to enable newly built and re-engineered applications for more effective business processes (for example, Supply Chain Management).
- You can make mergers and acquisitions a success by integrating dissimilar IT infrastructures from more than one company so that they can work together as a single entity.
- You can react more quickly to market trends and opportunities because your IT systems are flexible and dependable, and no longer constraining.
- You can overcome the barriers of diverse computer systems, geographic boundaries, time differences, language and format differences, and different methods of working.

WebSphere MQ messaging provides a secure and far-reaching communications infrastructure that you can expand with WebSphere Message Broker to apply intelligence to your business data as it travels through your network.

Transports
The main components of WebSphere Message Broker (the broker, the Configuration Manager, the User Name Server, and the Message Broker Toolkit) communicate using WebSphere MQ's communications protocol, WebSphere MQ Enterprise Transport. Your business applications, which can be on any of more than thirty industry platforms including those from IBM, Microsoft, and Sun Microsystems, Inc., can connect to the broker using WebSphere MQ protocols, or using other supported protocols which include WebSphere MQ Mobile Transport,, WebSphere MQ Real-time Transport, WebSphere MQ Multicast Transport, WebSphere MQ Web Services Transport, WebSphere Broker JMS Transport, SOAP/HTTP and TCP/IP.

The benefit of using WebSphere MQ protocols (WebSphere MQ Enterprise Transport or WebSphere MQ Mobile Transport) is that they provide assured, once-only delivery of messages between the components.
WebSphere MQ protocols provide rich support for applications:

- The Message Queue Interface (MQI) and Application Messaging Interface (AMI) are supported in several programming languages.
- The point-to-point (including request/reply and client/server) and publish/subscribe application communication models are supported.
- The complexities of communications programming are handled by the messaging services and are therefore removed from the application logic.
- The applications can access other systems and interfaces through adapters and gateways to products such as Lotus® Domino, Microsoft Exchange/Outlook, SAP/R3, and CICS and IMS/ESA products.

WebSphere Message Broker
WebSphere Message Broker provides a powerful message broker solution driven by business rules. Messages are formed, routed, and transformed according to the rules defined by an easy-to-use graphical user interface (GUI).
Diverse applications can exchange information in dissimilar forms, with brokers handling the processing required for the information to arrive in the right place in the correct format, according to the rules that you have defined. The applications do not need to know anything except their own conventions and requirements.
Applications also have much greater flexibility in selecting which messages they want to receive, because they can specify a topic filter, or a content-based filter, or both, to control the messages that are made available to them.
WebSphere Message Broker provides a framework that supports supplied, basic, functions along with user-defined enhancements, to enable rapid construction and modification of business processing rules that are applied to messages in the system.

Using WebSphere Message Broker in your business
WebSphere Message Broker addresses the needs of business and application integration by managing the flow of information. It provides services, based on message brokers, to allow you to:

- Route a message to several destinations, using rules that act on the contents of one or more of the fields in the message or message header.
- Transform a message, so that applications using different formats can exchange messages in their own formats.
- Store a message, or part of a message, in a database.
- Retrieve a message, or part of a message, from a database.
- Modify the contents of a message; for example, by adding data extracted from a database.

- Publish a message to make it available to other applications. Other applications can choose to receive publications that relate to specific topics, or that have specific content, or both.
- Create structured topic names, topic-based access control functions, content-based subscriptions, and subscription points.
- Exploit a public interface to develop message processing node types that can be incorporated into the broker framework to complement or replace the supplied nodes, or to incorporate node types developed by Independent Software Vendors (ISVs).
- Enable instrumentation by products such as those developed by Tivoli®, using system management hooks.

The benefits of WebSphere Message Broker can be realized both within and outside your enterprise:

- Your processes and applications can be integrated by providing message and data transformations in a single place, the broker. This integration helps to reduce the cost of application upgrades and modifications.
- You can extend your systems to reach your suppliers and customers, by meeting their interface requirements within your brokers. This ability can help you to improve the quality of your interactions, and allow you to respond more quickly to changing or additional requirements.

How To Do a WMB Sizing

When you select WMB sizing support from the ***Product Selection*** window to begin a new sizing, the ***WMB Application Definition*** input window is presented.

The general approach to sizing WMB applications on System z is to determine which of the pre-defined scenario profiles best represent the customer's application and provide an average scenario execution rate per second during a peak interval for each of these. Refer to the *WMB Application Definition* window to see the list of scenario profiles that can be included in a sizing. The **Profile** button for each scenario can be used to display a detailed description of the processing included in that scenario. In addition to the scenario execution rate, also provide the message persistence and message size. Default values are provided for each of these input fields.

The usual span of time for a peak interval is 15 minutes, and you want to specify the average transactions per 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 see results, click on the **Summary Report**, **CPU Utilization** or **Transaction Rate** buttons in the Reports and Capacity Projections section of the window.

The application window images shown in this user guide have been provided as a representation of the windows the user will see when using **zPSG** but there may be minor differences from the current version of **zPSG,** such as version numbers and dates.

WMB Scenario Profiles

Below are the detailed descriptions of the pre-defined scenario profiles supported in this tool. A scenario is included in a sizing by providing a scenario execution rate per second greater than zero. The message size and message persistence should also be specified if applicable.

Message Routing

The message routing use case shows how a database table can be used to store routing information which a message flow can then use to route messages to WebSphere MQSeries queues.

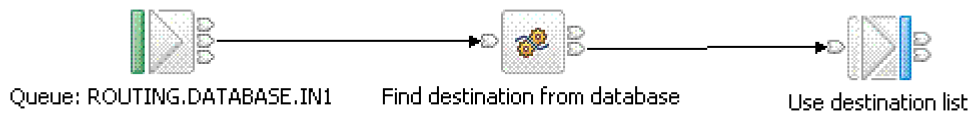
The message routing use case shows how to implement a routing table, using shared variables, to route messages in a message flow. Two versions of the message flow were used in these evaluations. One using a database was run as the WebSphere Business Integration Message Broker V5 test case and the second using the routing table implemented using shared variables was run as the WebSphere Message Broker V6 test case.

The processing in the message flows is described below:

Routing_using_database_table Message Flow

The message flow performs the following processing:

1. Reads a WebSphere MQ message containing an XML payload under transactional control.
2. Creates a destination list based on data in a database table and then routes the message to the entries in the destination list. Note this involves a read to the database for every message processed.
3. Produces a WebSphere MQ output message. The destination of the message is specified in the destination list.

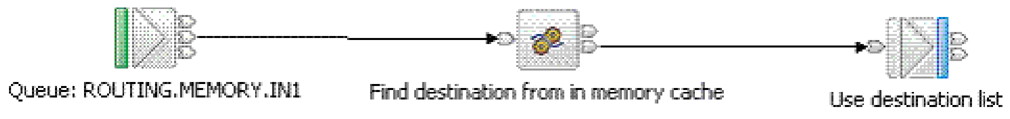


This version of the message flow was used for the WebSphere Business Integration Message Broker V5 measurements.

Routing_using_memory_cache Message Flow

The message flow performs the following processing:

1. Reads a WebSphere MQ message containing an XML payload under transactional control.
2. Creates a destination list based on data which is held in shared variables.
3. Produces a WebSphere MQ output message. The destination of the message is specified in the destination list.


Queue: ROUTING.MEMORY.IN1
Find destination from in memory cache
Use destination list

Further information about the Message Routing use case can be found in the Message Brokers section of the Technology samples category which is in the samples gallery of the WebSphere Message Broker development toolkit.

Transformation using ESQL

The transformation using ESQL use case is based on processing of sales data. At the time of sale the customer name, the code for the product, a description of the product, its category, the unit price and quantity purchased are recorded. Each customer may purchase several items.

Subsequently a statement is produced for each customer and it is the production of the statement that is performed in this use case. The processing results in a restructuring of the original message.

The messages used (input and output) are self-defining XML messages. Each message with sales data consists of three parts:

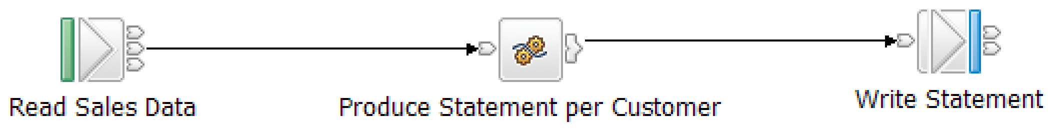
- A header containing a count of the number of repetitions of the repeating SaleList structure that follows.
- The body that contains the repetitions of the repeating SaleList structure.
- The trailer that contains the time the message was processed.

The production of the statement for each customer within a SaleList is achieved with a single message flow, the Transformation with ESQL Message Flow.

Transformation with ESQL Message Flow

The message flow performs the following processing:

1. Reads a WebSphere MQ message containing an XML payload under transactional control.
2. The input message is parsed and an invoice produced for each customer. This is achieved with a single Compute node containing ESQL.
3. Produces a WebSphere MQ output message containing an XML payload under transactional control.



Further information about the Transformation with ESQL use case can be found in the Message Brokers performance reports (http://www.ibm.com/support/docview.wss?rs=171&uid=swg27007150&loc=en_US&cs=utf-8&lang=en)

Coordinated Request/Reply

The coordinated request reply use case is based on the scenario of a contemporary and established application communicating through the use of WebSphere MQ messages in a request/reply processing pattern. The contemporary application uses self-defining XML messages and issues a request message. The established application uses Custom Wire Format (CWF) messages. It receives a request message, processes it and delivers a reply message. For the applications to successfully communicate, the message formats must be transformed for both the request and reply messages.

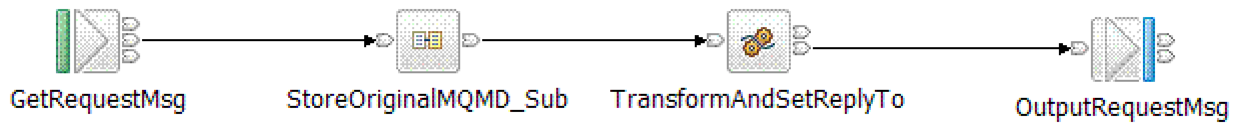
The processing in the use case consists of three message flows and one message set. The message flows are:

Request Message Flow

The request message flow performs the following processing:

- Reads a WebSphere MQ message containing an XML payload.
- Converts the message into the equivalent CWF format.
- Creates a WebSphere MQ message containing the transformed message.
- Saves the original ReplyToQ and ReplyToQMgr details in a separate WebSphere MQ message for subsequent retrieval by the Reply message flow.
- Sets the ReplyToQ and ReplyToQMgr details to be the input of the Reply message flow.
- Sends the message on to the Backend Reply message flow.

The Request message flow consists of the following nodes:

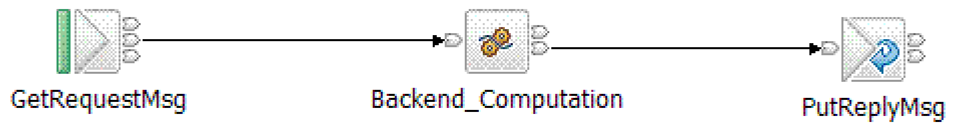


Backend Reply Message Flow

The backend reply message flows performs the following processing:

- Reads a WebSphere MQ message.
- Adds the time the message was modified to the payload of the message.
- Writes a WebSphere MQ message.

The Backend Reply message flow consists of the following nodes:



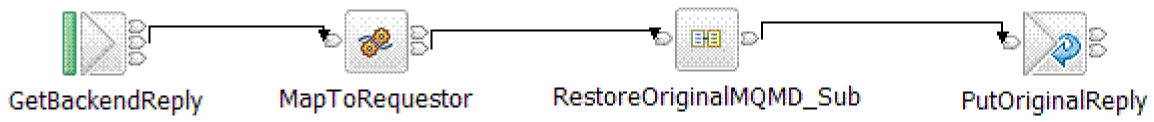
Reply Message Flow

The reply message flow performs the following processing:

1. Reads a WebSphere MQ message containing a message in CWF format.
2. Converts the message into the equivalent XML format.

3. Obtains the ReplyToQ and ReplyToQ Mgr of the original request message by reading the WebSphere MQ message which was used to store this information in the Request message flow. This is done by using the MQGET node.
4. Creates a WebSphere MQ message containing the transformed message and the retrieved ReplyToQ and ReplyToQMgr values.

The Reply message flow consists of the following nodes:



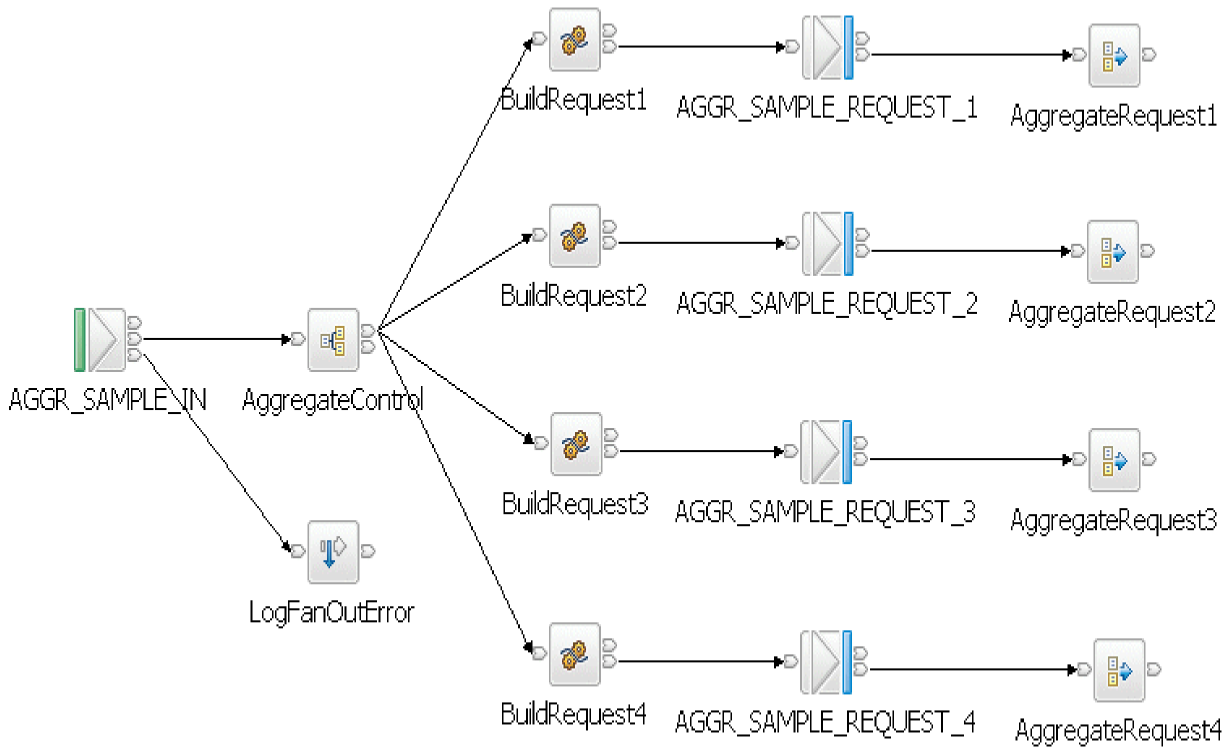
Further information about the Coordinated Request Reply use case can be found in the Message Brokers section of the Technology samples category which is in the samples gallery of the WebSphere Message Broker development toolkit.

Aggregation

The Aggregation use case demonstrates a simple four-way aggregation operation, using the Aggregate Control, Request, and Reply nodes. It contains three message flows to implement a four-way aggregation: FanOut, RequestReplyApp, and FanIn. This is the type of processing that might be used to invoke four different applications to process a travel booking, one to organise each of the flight, hotel, car and money.

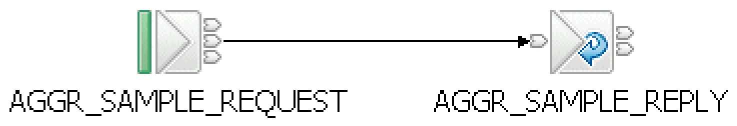
FanOut Message Flow

This is the flow that takes the incoming request message, generates four different request messages, sends them out on request/reply, and starts the tracking of the aggregation operation:

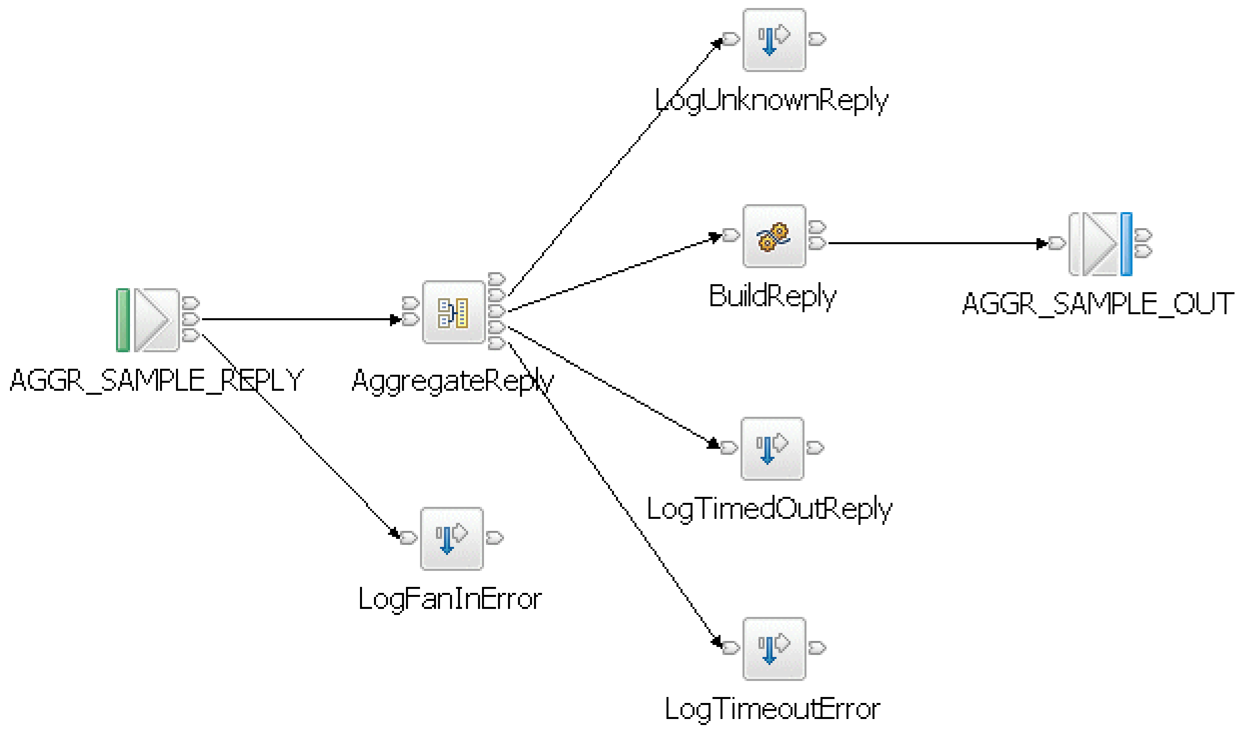


RequestReplyApp Message Flow

This message flow simulates the back-end service applications that would normally process the request messages from the aggregation operation. In a real system, these could be other message flows or existing applications. This message flow reads from the same queue that the MQOutput nodes in the FanOut flow write to, and it outputs to the queue that the input node which the FanIn flow reads from - it provides a messaging bridge between the two flows. The messages are put to their reply-to queue (as set by the MQOutput nodes in the FanOut flow).



FanIn Message Flow
This flow receives all the replies from the RequestReplyApp flow, and aggregates them into a single output message. The output message from the Aggregate Reply node cannot be output directly by an MQOutput node without some processing so a Compute node is added to process the data into a format where it can be written out to a queue.



Further information about the Aggregation use case can be found in the Message Brokers section of the Technology samples category which is in the samples gallery of the WebSphere Message Broker development toolkit.

Web Services – SOAP Nodes

The SOAP Nodes use case shows how the SOAP Input, Reply and Request nodes can be used to both provide and consume a Web Service.

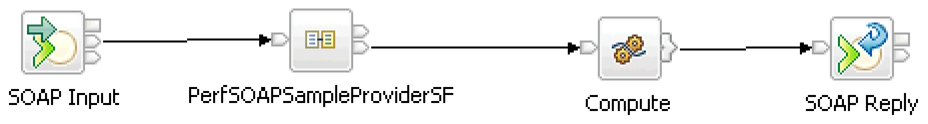
The starting point for the use case is a WSDL file that defines a simplified ordering service. The web service always returns a response indicating the part order is available; an option for extending the web service might be to use a Database node to query a stock database.

The Message Flows

This use case uses two message flows. One provides a Web Service and the other consumes a Web Service. These are described below.

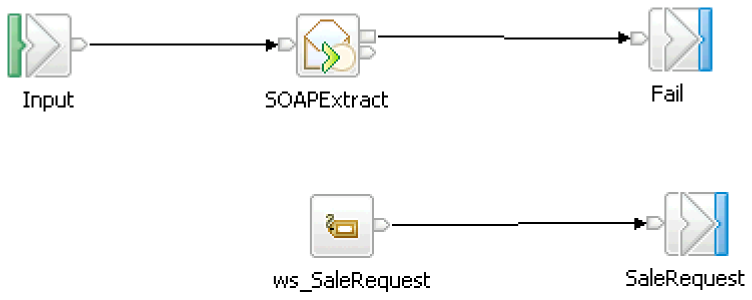
Web service provider message flow

The following figure shows the Web Service provider message flow:



The SOAP Input node recieves incoming SOAP messages and, if they are valid, passes them down the message flow to the Provider Extract subflow subflow which appears with the name PerfSOAPSampleProviderSF.

The Provider Extract subflow was created from the "Start from WSDL and/or XSD files" wizard in the Message Broker Toolkit and looks like this:



The SOAP Extract node takes a SOAP message and removes the SOAP envelope. In this use case, this leaves an XML message which can now be used in the XMLNSC domain in nodes such as the Mapping node or the Compute node.

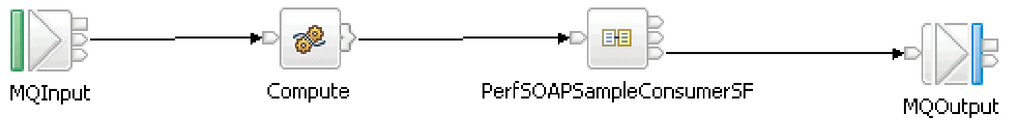
The SOAP Extract node then routes the message to a label based on the Web Service operation that is being invoked. In this use case, only one operation is used.

Upon exiting the subflow and returning to the main provider message flow, the XMLNSC message enters a Compute node in which an output message is created using minimal processing.

Processing then passes to the SOAP Reply node which constructs a suitable SOAP message to return to the Web Service consumer that initiated the Web Service call.

Web service consumer message flow

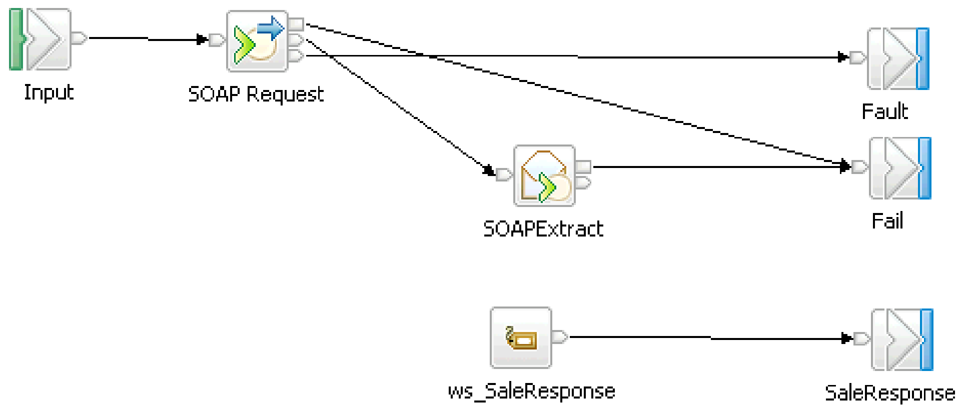
The following figure shows the Web Service consumer message flow:



This Web Service consumer flow is initiated by an MQ message arriving on the queue monitored by the MQ Input node. In this use case, this is an XML message in the XMLNSC domain. This message then enters a Compute node where the incoming data is used to create the XML data that is to be passed to Web Service.

Processing then enters the Comsumer subflow, shown as PerfSOAPSampleConsumerSF in the figure above.

The Consumer subflow was created by the "Start from WSDL and/or XSD files" wizard in the Message Broker Toolkit and looks like this:



The SOAP Request node takes the incoming XML data and uses it to build a valid SOAP message that which is then sent to the Web Service.

Assuming a valid response is recieved, it is passed to a SOAP Extract node which removes the SOAP envelope from the reponse, returning the message to one in the

XMLNSC domain. At this point the message is routed to the SaleResponse label and the subflow exits at which point control is returned to the main message flow.

At this point the message is sent to an MQ Output node which writes the XML data to the specified MQ queue.

Large Messaging

The Large Messaging use case is based on the end-of-day processing of sales data. Messages recording the details of sales through the day are batched together in the store for transmission to the IT centre. On receipt at the IT centre the batched messages are split back out into their constituent parts for subsequent processing.

This splitting is achieved using a WebSphere Message Broker message flow. Each of the individual messages representing a sale has the same structure.

The input and output messages in this use case are implemented as self-defining XML messages for simplicity. Other message formats could easily be used.

Each input message consists of three parts:

- A header containing a count of the number of repetitions of the repeating SaleList structure that follows.
- The body that contains the repetitions of the repeating SaleList structure.
- The trailer that contains the time the message was processed.

The aim of the processing in this use case is to write each of the instances of the SaleList structure as a separate WebSphere MQ message while minimizing overall memory requirements.

The message flow implements a memory saving technique through the use of a mutable message tree.

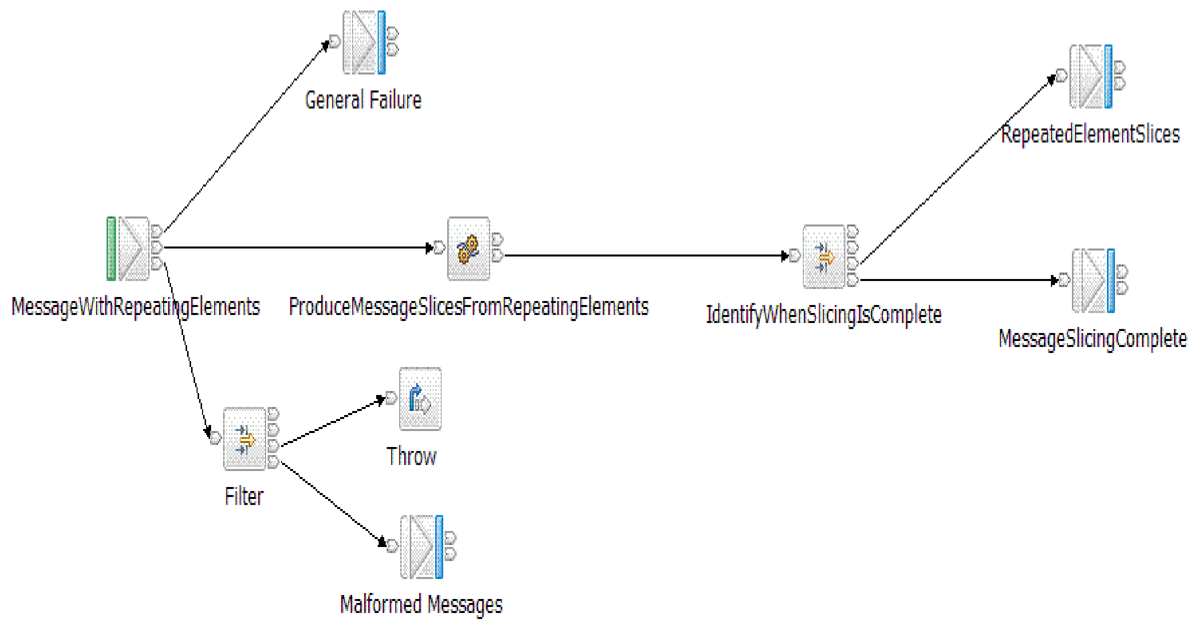
The processing in the use case consists of one message flow. The processing it performs is described below.

Large Messaging Message Flow

The large messaging message flow performs the following processing:

1. Reads a WebSphere MQ message containing an XML payload under transactional control.
2. Formats a WebSphere MQ message for each instance of the SaleList structure.
3. Writes the WebSphere MQ messages to the output queue.
4. Produces a WebSphere MQ message to signal completion of the processing when the final element has been processed.

The Large Messaging message flow consists of the following nodes:


General Failure
RepeatedElementSlices
MessageWithRepeatingElements
ProduceMessageSlicesFromRepeatingElements
IdentifyWhenSlicingIsComplete
MessageSlicingComplete
Throw
Filter
Malformed Messages

Further information about the Large Messaging use case can be found in the Message Brokers section of the Technology samples category which is in the samples gallery of the WebSphere Message Broker development toolkit.

Data Warehouse

The Data Warehouse use case demonstrates a scenario in which a message flow is used to perform the archiving of data, such as sales data, into a database. The data is stored for later analysis by another message flow or application.

Because the sales data is analyzed at a later date, the storage of the messages has been organized in a way that makes it easy to select records for specified times. The date and time at which the WebSphere MQ message containing the sales record was written are stored as separate column values when the message is inserted into the database. The database table contains four columns:

- The message data - the payload of the WebSphere MQ message stored as a BLOB.
- The date on which the WebSphere MQ message was created.
- The time when the WebSphere MQ message was created.
- A time stamp created by the database to record the time when the record was inserted.

By storing the data in this way it is possible to retrieve records between specific periods of time, say between the hours of 9:00 a.m. to 12:00 p.m. or 12:01 p.m. and 5:00 p.m. which would allow a comparison of morning and afternoon sales to be made.

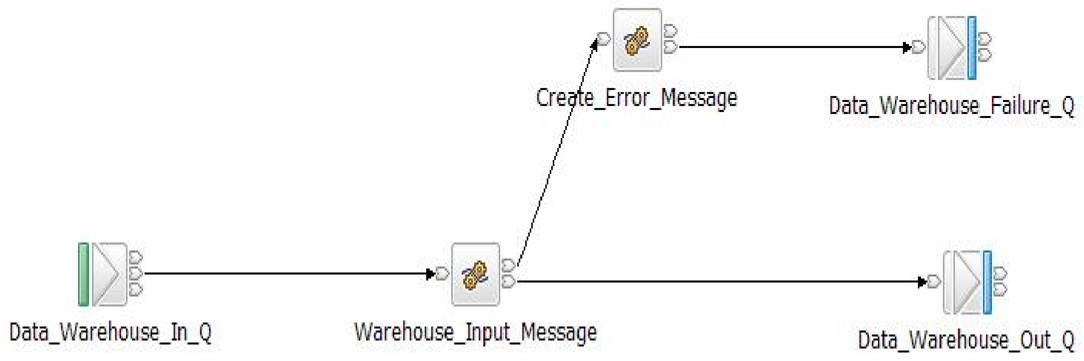
The data archiving is performed by the WarehouseData message flow. This is described below.

WarehouseData Message Flow

The WarehouseData message flow performs the following processing.

1. Reads a WebSphere MQ message containing an XML payload. The payload contains the data to be archived.
2. Converts a portion of the message tree to a BLOB ready for insertion into the database.
3. Inserts the message BLOB along with the date and time at which the WebSphere MQ message was written into a database.
4. Sends a WebSphere MQ confirmation message to signal successful insertion of the message into the database.

The WarehouseData message flow consists of the following nodes:



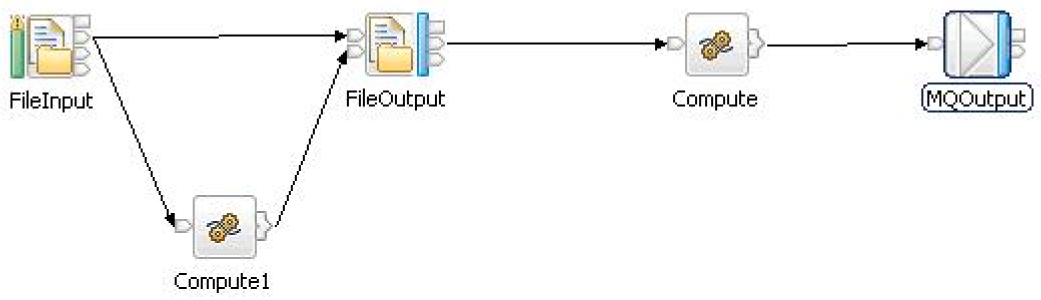
Further information about the Data Warehouse use case can be found in the Message Brokers section of the Technology samples category which is in the samples gallery of the WebSphere Message Broker development toolkit.

File Nodes

This use cases illustrates use of the file processing nodes that became available in Message Broker V6.1.

In order to be able to determine the processing rate it was necessary to insert an MQ Output node after the FileOutput Node so that the processing rate could be measured by the automation which invokes and measures the use cases. Otherwise it would have been necessary to have run an additional program on the server to monitor and count the files. This would have been a larger overhead than the alternatives and would have distorted the results.

This use case consists of a FileInput Node, Compute Node, FileOutput Node and an MQ Output Node. The nodes are shown in the figure below.



Each of the large files (each was ~100MB) read by the FileInput node consists of multiple records. The size of record varies for different executions of the use case although the file size stays constant. The size of the record determines the number of records in the file as the file was of a fixed size. For example with 4K records the 100MB file contains 256000 records. Each record is a valid XML document.

The output file is closed when it reaches 1GB of data regardless of record size. This is achieved by using the first compute node to count the number of records read and to check the record size. When 1GB of data was written it sent a message to the Finish File terminal of the FileOutput node.

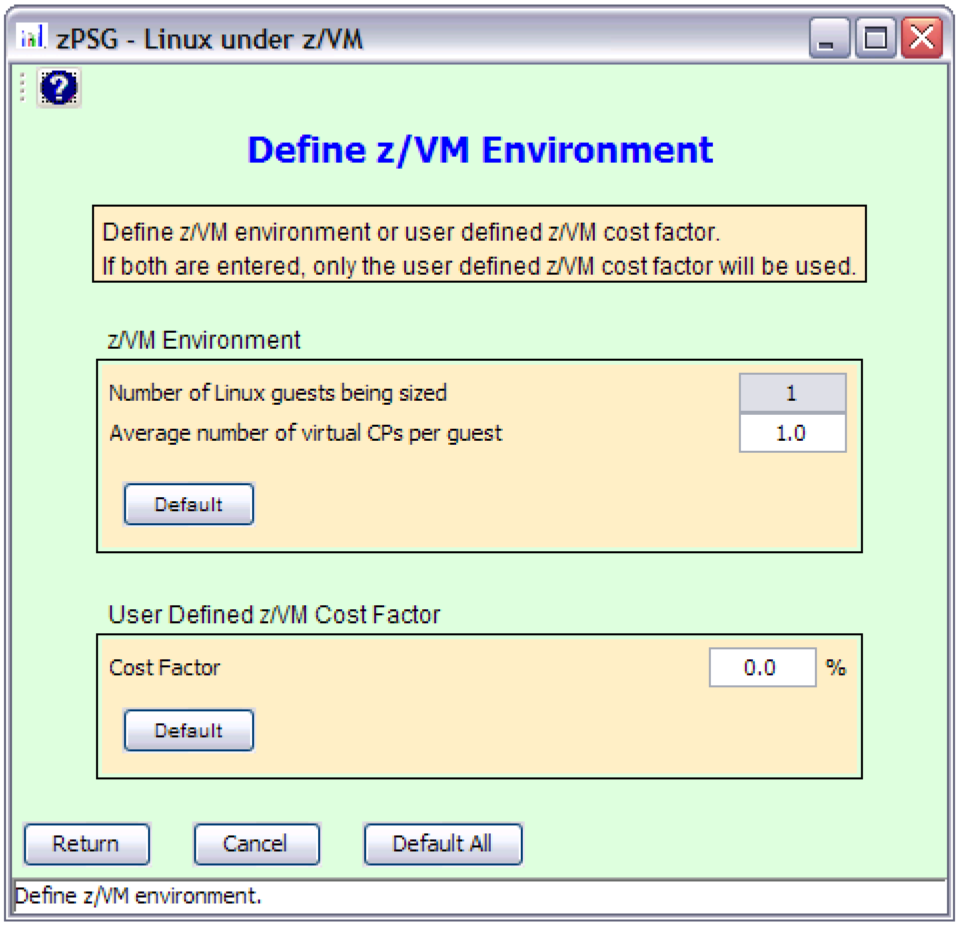
All records are processed using a message Domain of XMLNSC.

The FileInput node is configured to set the Record Detection property to Parsed Record Sequence using the XMLNSC parser. Input files are deleted once processed.

The File Output Node is configured to set the Record Definition to unmodified.

For every record written to the file an empty MQ message is propagated to the MQ Output node. The queue is drained by a client application and the message rate is reported. This rate represents the number of records being written to the output file per second.

Linux under z/VM



Define z/VM Environment

Define z/VM environment or user defined z/VM cost factor.
If both are entered, only the user defined z/VM cost factor will be used.

z/VM Environment

Number of Linux guests being sized	1
Average number of virtual CPs per guest	1.0

User Defined z/VM Cost Factor

Cost Factor	0.0 %

Define z/VM environment.

This window is displayed when the **Linux deployed as a guest under z/VM** button is clicked on the primary definition window for the application being sized.

When Linux is deployed as guest under z/VM the CPU capacity requirement for the workload increases based on a number of factors including the workload characteristics, number of Linux guests being sized, and the average number of virtual CPs assigned to each guest.

The user can define the z/VM environment and let the tool estimate the z/VM cost factor or they can specify the user defined z/VM cost factor. When both are entered, the user defined z/VM cost factor will be used to calculate the capacity requirement with z/VM.

z/VM Environment

Number of Linux guests being sized
This field is automatically assigned based on the number of guests being sized. When sizing a single application the value will always be 1. This number varies in an aggregation sizing based on the number of applications assigned to the concurrency set being sized. This field is display only and can't be changed by the user.

Average number of virtual CPs per guest
Average number of virtual CPs assigned to guest(s) being sized. Default value is 1.0

User Defined z/VM Cost Factor

Cost Factor
User defined z/VM cost factor specified as a percentage. Default value is 0.0%

Note: When Linux is deployed as a single guest under z/VM and the number of virtual CPs is equal to the number of real CPs, you should increase the CPU capacity requirement by 9% for WMB.

Push Buttons

Click the **Return** button to return to the calling window.

Click the **Default All** button to set all the values on this window to the default setting.

Click the **Cancel** button to return to the calling window without saving any changes.

When Linux is deployed under z/VM with multiple guests involved, a detailed z/VM sizing should be done using the **zVM-Planner** tool, described on the next page. The capacity requirement for each Linux guest, determined from **zPSG**, will be needed as an input metric. For sizing Linux guests with workload environments that are not yet supported in **zPSG**, contact Techline for sizing assistance.

zPSG application sizings are generally done for peak period activity. When many Linux guests are active under a single z/VM image, it is likely that the individual guest peaks do not occur at the same time. Therefore an opportunity exists for complementary peaks, thus lessening the overall z/VM capacity requirement. The **zVM-Planner** tool can help with this assessment.

You can also request sizing assistance for z/VM from Techline, using the appropriate URL:

For IBMers:	http://w3-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS2881
For BPs:	http://www.ibm.com/partnerworld/techline

z/VM Planner for Linux Guests on IBM System z Processors

zVM-Planner is a PC-based productivity tool under Windows XP or Vista, designed to provide capacity planning insight for IBM System z processors running various Linux applications as guests under z/VM. Capacity results are based on analysis of a variety of benchmarks, both Linux native and Linux under z/VM. The tool is generic concerning software release levels, generally applying to z/VM v5.1 and later.

zVM-Planner input consists primarily of VM guest definitions and capacity requirements for each intended Linux application (a variety of Linux applications are supported). The expected capacity requirement for each Linux guest (a required input) can usually be obtained using the companion **zPSG** sizing tool or Techline assistance. The combined guest capacity requirement is then determined for optimally complementary peaks and for totally concurrent peaks. The degree of peak concurrency is user-selectable between these values. The resulting capacity requirement is combined with that of VM to support the entire complement of guests. Another companion tool, **zPCR**, can then be used to identify a processor and partition that can accommodate the VM image. All capacity values must be relative to a Reference-CPU setting that is common for all tools involved.

For scenarios where new Linux guests are to be added to an existing VM image, a **zVM-Planner** model of the existing VM guest configuration should first be built. The new Linux guests can then be added to determine to overall VM capacity requirement.

For scenarios where a VM image is to be added to an existing host processor, a **zPCR** model of the existing LPAR configuration should first be built. A partition that can deliver the capacity required by the VM image can then be added. **zPCR** can help assess any processor upgrade that may be necessary to accommodate the VM image.

Several guest metrics are available to help balance how the overall capacity will be distributed, including the number of VCPs (virtual CPs), and Share and Capping assignments.

Results are presented in tables and graphs that can be captured for documentation purposes. **zVM-Planner** studies can be saved for future reference. Both a User's Guide and integrated context sensitive help are included.

IBM employees can obtain **zVM-Planner** and other CPS tools via the Intranet

w3.ibm.com/support/americas/wsc/cpsproducts.html

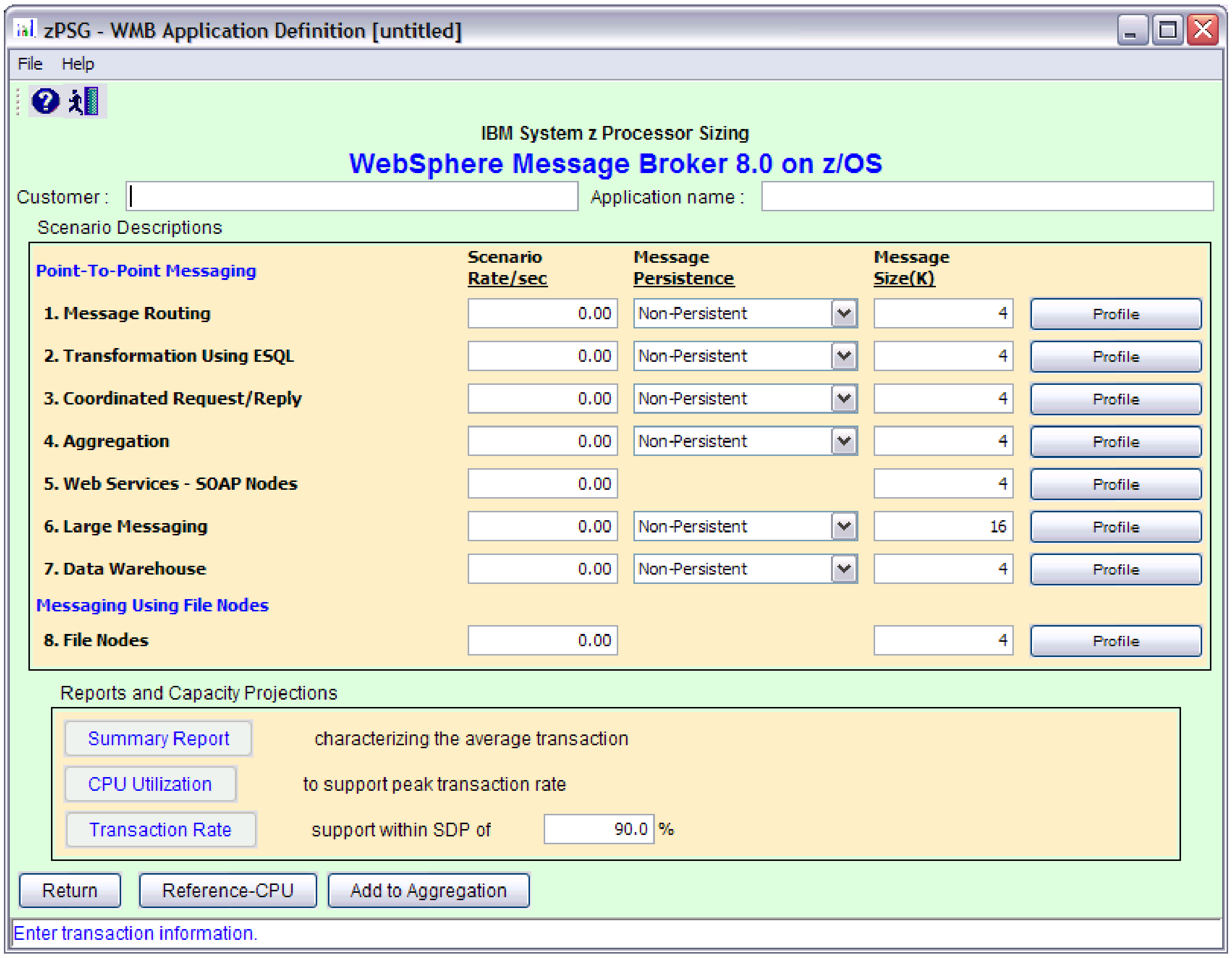
IBM Business-Partners can obtain **zVM-Planner** and other CPS tools via the Internet

https://www-304.ibm.com/partnerworld/wps/servlet/mem/ContentHandler/tech_PRS1762

For questions concerning CPS tools, contact Capacity Planning Support:

- Lotus Notes: CPS PC Tools/Gaithersburg/IBM@IBMUS
- E-mail: cpstools@us.ibm.com

WMB Application Definition



zPSG - WMB Application Definition [untitled]

File Help

IBM System z Processor Sizing

WebSphere Message Broker 8.0 on z/OS

Customer :
Application name :

Scenario Descriptions

Point-To-Point Messaging	Scenario Rate/sec	Message Persistence	Message Size(K)	
1. Message Routing	0.00	Non-Persistent	4	Profile
2. Transformation Using ESQL	0.00	Non-Persistent	4	Profile
3. Coordinated Request/Reply	0.00	Non-Persistent	4	Profile
4. Aggregation	0.00	Non-Persistent	4	Profile
5. Web Services - SOAP Nodes	0.00		4	Profile
6. Large Messaging	0.00	Non-Persistent	16	Profile
7. Data Warehouse	0.00	Non-Persistent	4	Profile
Messaging Using File Nodes				
8. File Nodes	0.00		4	Profile

Reports and Capacity Projections

Summary Report	characterizing the average transaction
CPU Utilization	to support peak transaction rate
Transaction Rate	support within SDP of 90.0 %

Return | Reference-CPU | Add to Aggregation

Enter transaction information.

This window is displayed when the <u>**PSG**</u> button is clicked on the ***Product Selection*** window when **WebSphere Message Broker** has been selected for z/OS or Linux (the example used here is for z/OS).

Note: A transaction rate greater than zero must be specified for at least one scenario profile to get a sizing estimate.

Please note that the WMB version supported for z/OS and Linux is 8.0.

Description of Input Fields

<u>Menu bar</u>

File

New	Start a new study. Sets all fields to initialization values.
Load	Load a previously saved study
Save	Save the current study
Save as	Save the current study as a new file
Exit (Ctrl-E)	Exit window and return to the ***Product Selection*** window
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

<u>Toolbar</u>

<u>?</u> button	Click this button to go to Help for this window.
Exit zPSG button	Click this button to terminate **zPSG** execution.

<u>Customer =</u>
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.

<u>Application name =</u>
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.

Input Fields and Buttons

Scenario Rate/sec

Specify an execution rate per second value for each of the scenarios that should be included in the sizing. Specify a value of zero if the scenario should not be included in the sizing. The default value for each of the scenarios is zero.

Message Persistence

Select from Non-Persistent or Persistent for the scenarios that should be included in the sizing that support message persistence. The default value for each of the scenarios that support persistence is Non-Persistent.

Message Size (K)

Specify the average message size in KB for each of the scenarios that should be included in the sizing. For the File Node scenario, this value represents the file record size in KB. The default value for each of the scenarios is 4K except for the Large Messaging scenario which is 16K.

Profile button

Click this button to view a detailed description of the processing for the scenario associated with this button. Each scenario has a Profile button.

Note: The short description for each scenario is displayed in a tool tip when the mouse pointer is over the name of the scenario.

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. The **Linux under z/VM** button will be enabled and 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 transaction profiles included in the sizing.

CPU Utilization button

Click this button to see an output window with estimates of processor utilization for all System z processors supported in zPSG.

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 supported in zPSG. 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

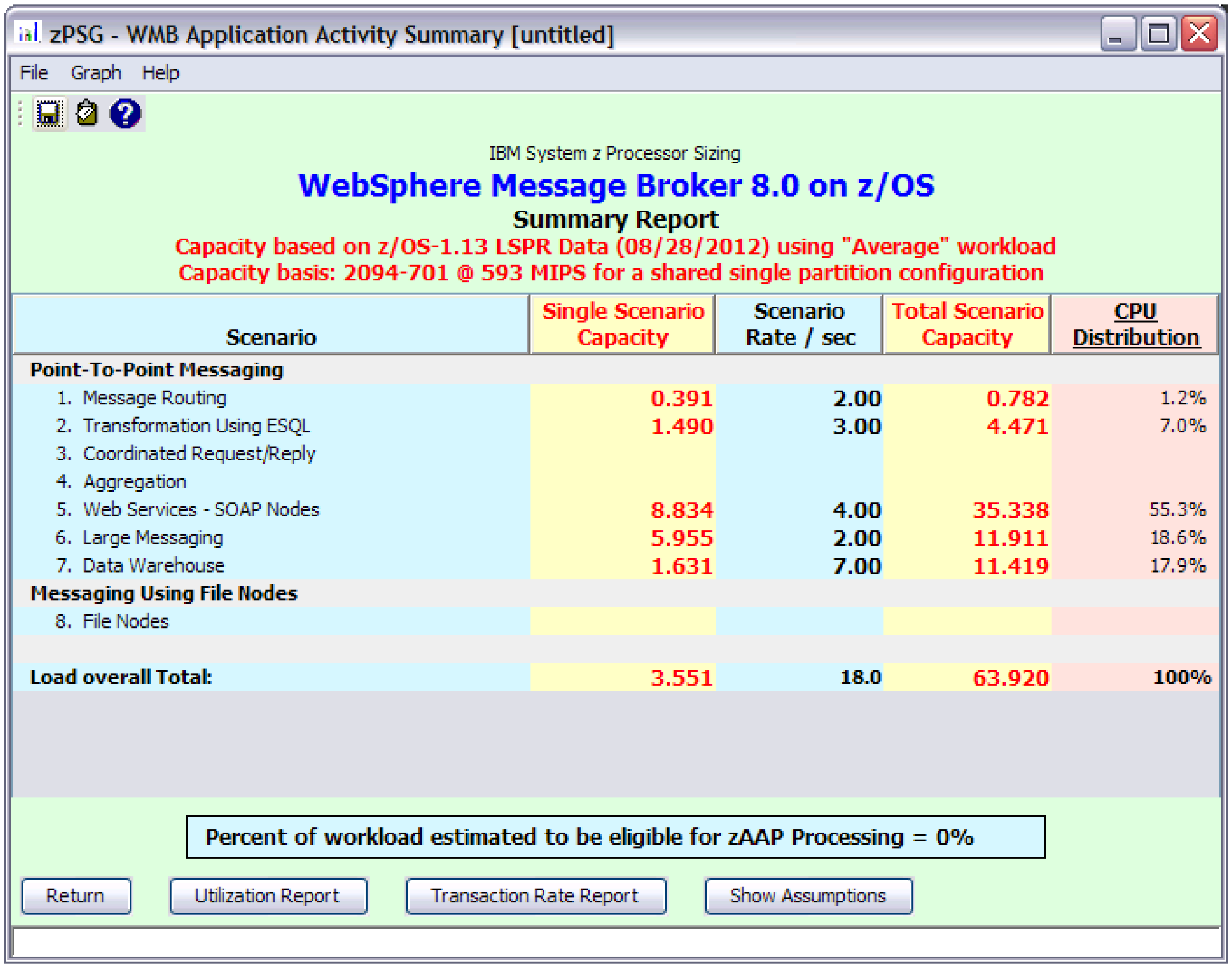
Click this button to return to the ***Product Selection*** window.

Reference-CPU button

Click this button to go to a window to change the System z processor used as a basis for capacity ratings. See the Reference-CPU section in the zPSG User’s Guide for information about this setting.

Linux under z/VM button **For Linux only**
Click this button for important sizing considerations when Linux is deployed as a guest under z/VM. See Linux under z/VM for details about these sizing considerations.

WMB Application Activity Summary



IBM System z Processor Sizing

WebSphere Message Broker 8.0 on z/OS

Summary Report

Capacity based on z/OS-1.13 LSPR Data (08/28/2012) using "Average" workload
Capacity basis: 2094-701 @ 593 MIPS for a shared single partition configuration

Scenario	Single Scenario Capacity	Scenario Rate / sec	Total Scenario Capacity	CPU Distribution
Point-To-Point Messaging				
1. Message Routing	0.391	2.00	0.782	1.2%
2. Transformation Using ESQL	1.490	3.00	4.471	7.0%
3. Coordinated Request/Reply				
4. Aggregation				
5. Web Services - SOAP Nodes	8.834	4.00	35.338	55.3%
6. Large Messaging	5.955	2.00	11.911	18.6%
7. Data Warehouse	1.631	7.00	11.419	17.9%
Messaging Using File Nodes				
8. File Nodes				
Load overall Total:	3.551	18.0	63.920	100%

Percent of workload estimated to be eligible for zAAP Processing = 0%

This window is displayed when the **Summary Report** button is clicked on the primary ***WMB Application Definition*** window. It shows a breakdown of the CPU per transaction for the various pre-defined transactions included in the sizing.

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.

Scenario **column**

Lists the pre-defined scenarios available on the primary ***WMB Application Definition*** window.

Single Scenario Capacity **column**

Reflects the amount of CPU (as represented by the Capacity Rating) for each scenario, and at the bottom for all transactions.

Scenario Rate / sec **column**

Reflects the number of scenarios completed per second for each of the pre-defined scenarios as specified on the ***WMB Application Definition*** window.

Total Tran Capacity **column**

Reflects the amount of CPU (as represented by the Capacity Rating) for each scenario multiplied by the transaction rate specified in the ***Scenario Rate / sec*** column, and at the bottom for all transactions.

CPU Distribution **column**

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

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

Shows the estimated percentage of Java content for the pre-defined scenarios included in the sizing. Percentages of Java content were computed in all the performance lab measurements done to support the pre-defined scenarios supported in this tool. Depending on what pre-defined scenarios you included, 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.

Push Buttons

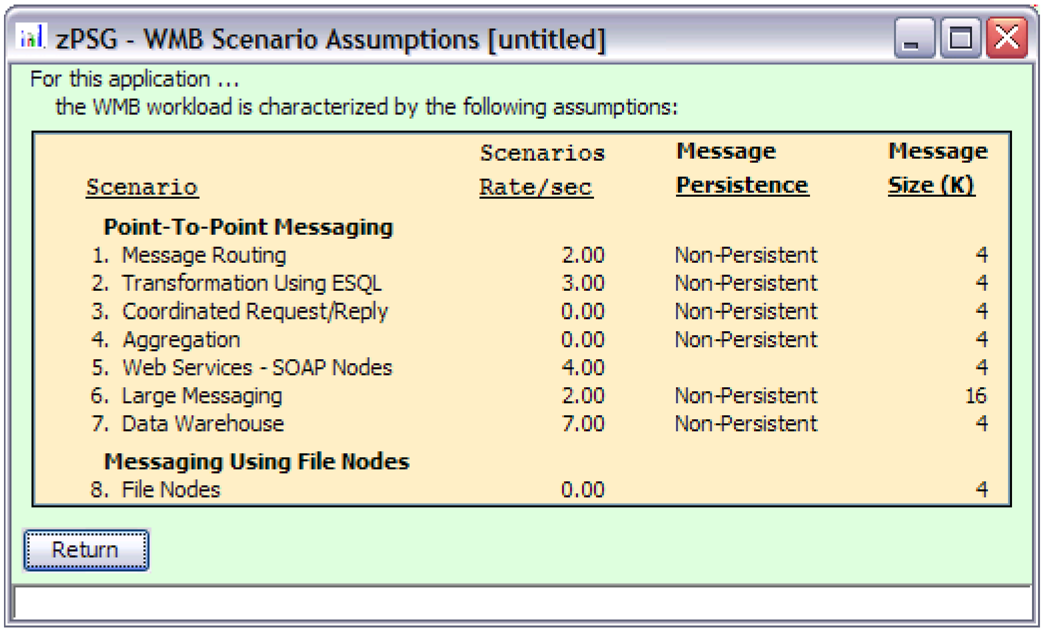
Click the **Return** button to return to the primary ***WMB Application Definition*** input window.

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

Click the **Transaction Rate Report** button to go to the ***WMB 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 ***WMB Application Transaction Assumptions*** window.

WMB Transaction Assumptions



zPSG - WMB Scenario Assumptions [untitled]

For this application ...
the WMB workload is characterized by the following assumptions:

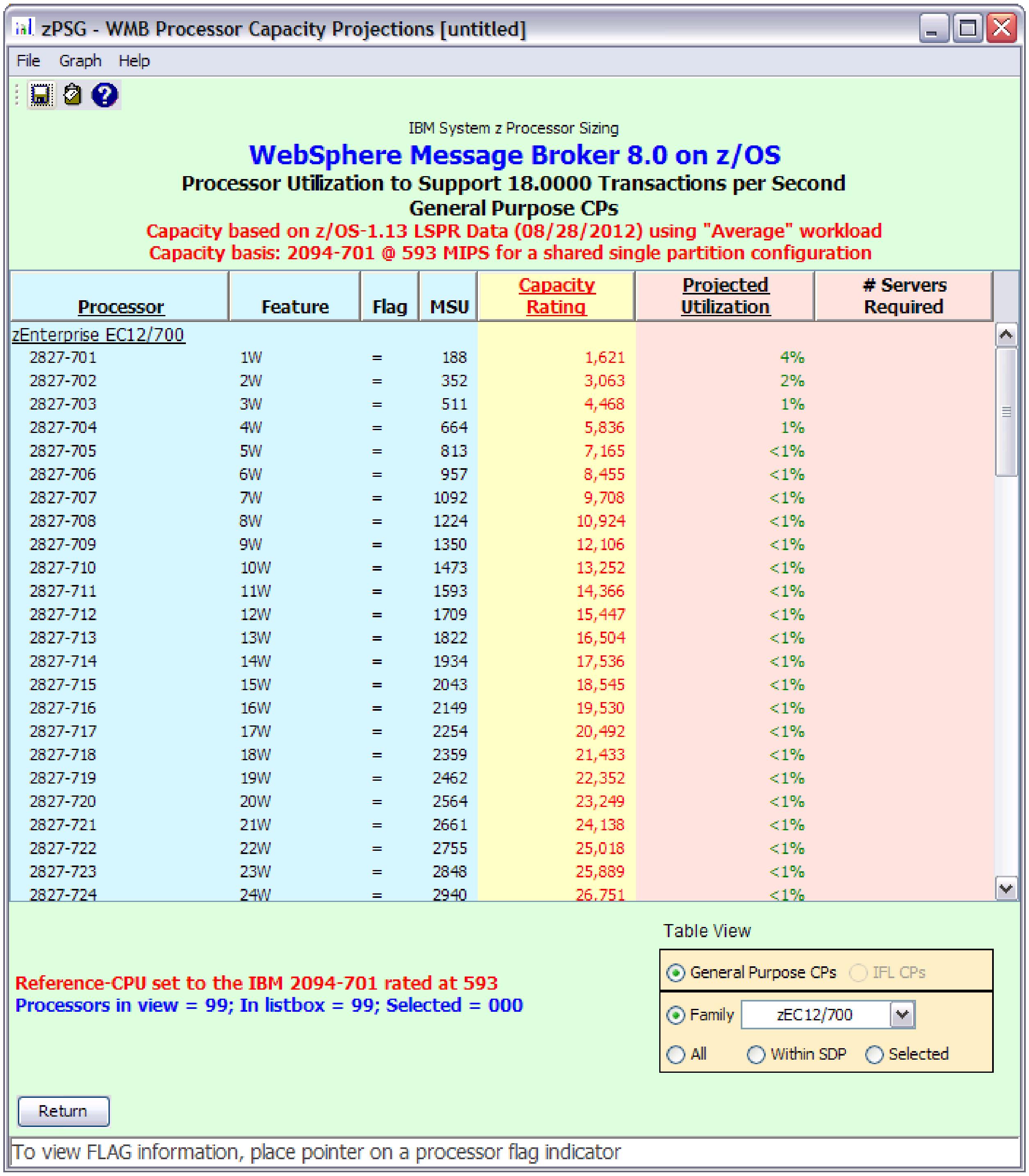
Scenario	Scenarios Rate/sec	Message Persistence	Message Size (K)
Point-To-Point Messaging			
1. Message Routing	2.00	Non-Persistent	4
2. Transformation Using ESQL	3.00	Non-Persistent	4
3. Coordinated Request/Reply	0.00	Non-Persistent	4
4. Aggregation	0.00	Non-Persistent	4
5. Web Services - SOAP Nodes	4.00		4
6. Large Messaging	2.00	Non-Persistent	16
7. Data Warehouse	7.00	Non-Persistent	4
Messaging Using File Nodes			
8. File Nodes	0.00		4

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

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

WMB Processor Utilization

WMB Processor Capacity Projections



IBM System z Processor Sizing

WebSphere Message Broker 8.0 on z/OS

Processor Utilization to Support 18.0000 Transactions per Second

General Purpose CPs

Capacity based on z/OS-1.13 LSPR Data (08/28/2012) 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	4%	
2827-702	2W	=	352	3,063	2%	
2827-703	3W	=	511	4,468	1%	
2827-704	4W	=	664	5,836	1%	
2827-705	5W	=	813	7,165	<1%	
2827-706	6W	=	957	8,455	<1%	
2827-707	7W	=	1092	9,708	<1%	
2827-708	8W	=	1224	10,924	<1%	
2827-709	9W	=	1350	12,106	<1%	
2827-710	10W	=	1473	13,252	<1%	
2827-711	11W	=	1593	14,366	<1%	
2827-712	12W	=	1709	15,447	<1%	
2827-713	13W	=	1822	16,504	<1%	
2827-714	14W	=	1934	17,536	<1%	
2827-715	15W	=	2043	18,545	<1%	
2827-716	16W	=	2149	19,530	<1%	
2827-717	17W	=	2254	20,492	<1%	
2827-718	18W	=	2359	21,433	<1%	
2827-719	19W	=	2462	22,352	<1%	
2827-720	20W	=	2564	23,249	<1%	
2827-721	21W	=	2661	24,138	<1%	
2827-722	22W	=	2755	25,018	<1%	
2827-723	23W	=	2848	25,889	<1%	
2827-724	24W	=	2940	26,751	<1%	

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

Processors in view = 99; In listbox = 99; Selected = 000

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

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

This window is displayed when the **CPU Utilization** button is clicked on the ***WMB Application Definition*** window or the **Utilization Report** button is clicked on the ***WMB 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 rate(s) and input parameters specified for the pre-defined transactions. 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:

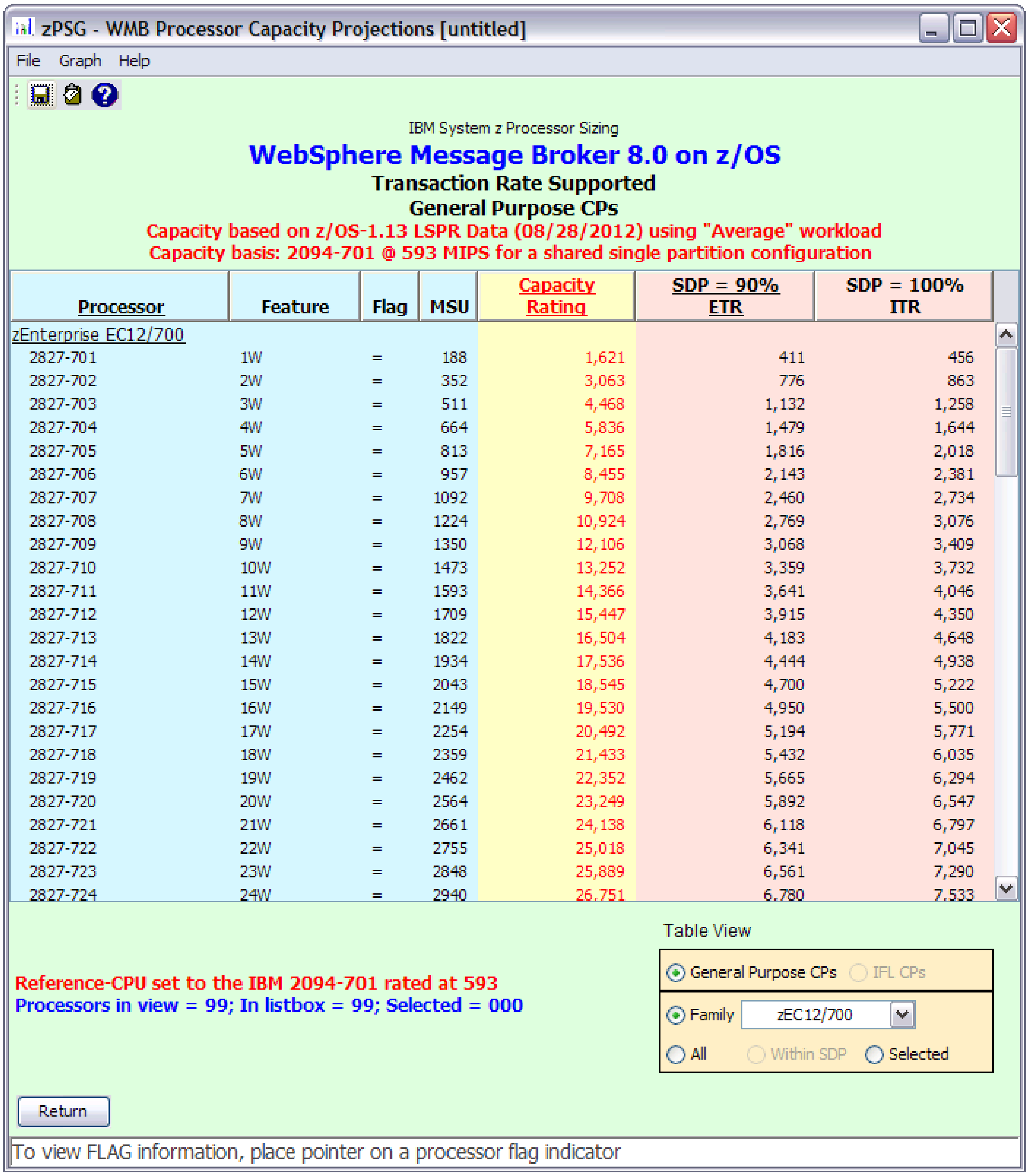
- **General Purpose CPs** shows entries with some number of general CP engines
- **IFL CPs** shows entries with some number of IFL engines (for Linux only)
- **Family** shows all processor models for the family selected (Default)
- **All** shows all processor models supported in zPSG
- **Within SDP** 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 ***WMB Application Definition*** window.

WMB Transaction Rate Supported

WMB Processor Capacity Projections



zPSG - WMB Processor Capacity Projections [untitled]

IBM System z Processor Sizing

WebSphere Message Broker 8.0 on z/OS

Transaction Rate Supported

General Purpose CPs

Capacity based on z/OS-1.13 LSPR Data (08/28/2012) 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	411	456
2827-702	2W	=	352	3,063	776	863
2827-703	3W	=	511	4,468	1,132	1,258
2827-704	4W	=	664	5,836	1,479	1,644
2827-705	5W	=	813	7,165	1,816	2,018
2827-706	6W	=	957	8,455	2,143	2,381
2827-707	7W	=	1092	9,708	2,460	2,734
2827-708	8W	=	1224	10,924	2,769	3,076
2827-709	9W	=	1350	12,106	3,068	3,409
2827-710	10W	=	1473	13,252	3,359	3,732
2827-711	11W	=	1593	14,366	3,641	4,046
2827-712	12W	=	1709	15,447	3,915	4,350
2827-713	13W	=	1822	16,504	4,183	4,648
2827-714	14W	=	1934	17,536	4,444	4,938
2827-715	15W	=	2043	18,545	4,700	5,222
2827-716	16W	=	2149	19,530	4,950	5,500
2827-717	17W	=	2254	20,492	5,194	5,771
2827-718	18W	=	2359	21,433	5,432	6,035
2827-719	19W	=	2462	22,352	5,665	6,294
2827-720	20W	=	2564	23,249	5,892	6,547
2827-721	21W	=	2661	24,138	6,118	6,797
2827-722	22W	=	2755	25,018	6,341	7,045
2827-723	23W	=	2848	25,889	6,561	7,290
2827-724	24W	=	2940	26,751	6,780	7,533

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

Processors in view = 99; In listbox = 99; Selected = 000

Table View

[✓ General Purpose CPs] IFL CPs

[✓ Family] zEC12/700

All Within SDP Selected

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

This window is displayed when the **Transaction Rate** button is clicked on the ***WMB Application Definition*** window or the **Transaction Rate Report** button is clicked on the ***WMB 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 ***WMB 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).

<u>Table View Options Box</u>

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 CP engines
- **IFL CP's** 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.

<u>Return</u> button

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

WMB Sizing Assistance

Here are instructions for accessing the System z questionnaire and submitting WMB 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 WMB 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 WMB 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.

WMB 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.

CTG
CICS Transaction Gateway, the IBM product providing JCA (J2EE) data connector support from WAS to CICS

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 Connect is used in our performance measurements to access DB2 on z/OS from WAS on Linux.

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.

RC4/MD5
The most commonly used SSL cipher and hashing algorithm. With RC4/MD5, System z9 and zSeries crypto hardware can be used for full (non-cached) handshakes, but not for the encryption & decryption of data.

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.