



IBM Software Group

IBM® WebSphere® Extended Deployment V6.1

WebSphere Virtual Enterprise

Formerly Operations Optimization

Configuring dynamic operations



@business on demand.

© 2007 IBM Corporation
Updated June 16, 2008

This presentation will describe how you configure the dynamic operations features of WebSphere Extended Deployment.

Agenda

- Creating runtime resources
 - ▶ Dynamic cluster
 - ▶ On demand router
- Configuring operational policies
 - ▶ Service policy
 - ▶ Work class
 - ▶ Classification rules
- Configuring autonomic managers
 - ▶ Application placement controller
 - ▶ Autonomic request flow manager
- Summary



The first topic covers the creation of runtime resources necessary for dynamic operations, including dynamic clusters and on demand routers. This is followed by configuration of operational policies, which define the work types and goal levels used to drive a dynamic operations environment. Finally this presentation will cover configuration options for the autonomic managers that are primarily responsible for dynamic operations.

Section

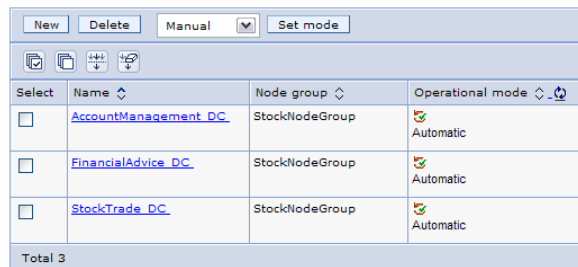
Creating runtime resources



This section will cover creating runtime resources.

Configuring dynamic clusters

- Dynamic clusters are created and modified under Servers → Dynamic clusters
- The main panel enables the creation or deletion of dynamic clusters and operational modes to be set
 - ▶ Manual, supervised, or automatic modes



Select	Name	Node group	Operational mode
<input type="checkbox"/>	AccountManagement_DC	StockNodeGroup	Automatic
<input type="checkbox"/>	FinancialAdvice_DC	StockNodeGroup	Automatic
<input type="checkbox"/>	StockTrade_DC	StockNodeGroup	Automatic

Total 3



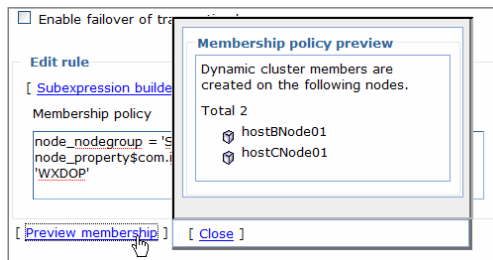
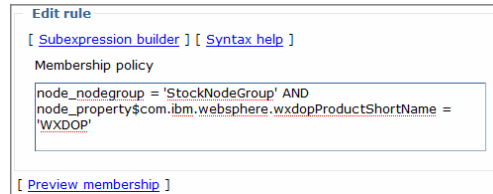
A dynamic cluster is similar to the familiar concept of a 'cluster' from WebSphere Application Server, but can be resized dynamically at run time. As demand for applications running on a dynamic cluster increases or decreases, instances of that dynamic cluster can be started or stopped on nodes within the cluster to accommodate the changes in load.

To enable this dynamic behavior, put the dynamic cluster into automatic mode by selecting the check box for the cluster. Next, select 'Automatic' from the pop-up menu, and click 'Set mode' on the screen shown here

A dynamic cluster can consist of WebSphere servers or other middleware servers. However, all servers in a dynamic cluster must be of the same type.

Dynamic cluster membership

- A membership method determines which servers can join the dynamic cluster
- Create an expression in the console
- Node group, node name, host name, or node property



- Each node is evaluated against the expression
 - ▶ If it matches it is considered a candidate to host a cluster member
- Preview membership allows you to verify your rule



When you create a dynamic cluster, you must specify which nodes can host members of the dynamic cluster. WebSphere Extended Deployment version 6.1 allows two methods to determine which nodes can join the dynamic cluster. You can identify pre-existing servers as members of your dynamic cluster or you can specify a “membership policy”, which is a rule that specifies the set of nodes that can host dynamic cluster members.

You can define membership rules based on node name, node host name, node property values and node group membership. You can create complex rules using the Boolean operators AND, OR, and NOT.

The membership policy is evaluated against the nodes in your cell when the dynamic cluster is created. WebSphere Extended Deployment will create servers for the dynamic cluster using nodes that match the membership policy that you define. If new nodes are added to your environment, they will automatically be added to the dynamic cluster if they match the defined membership policy. Similarly, if you change a membership policy, it is reevaluated and server instances are created or removed based on the new policy definition. If you change a node’s properties such that it should be added to or removed from an existing dynamic cluster the corresponding server instances are added or removed on that node when you save your changes.

“Preview membership” will evaluate the current rule and display a list of nodes that match the policy.

Dynamic cluster limits

- Specify requirements on number of running instances
 - ▶ Minimum number
 - Default is one.
 - ▶ Maximum instances
 - Default is number of nodes * vertical stacking factor
- Stop all instances

Minimum number of cluster instances

Stop all instances during periods of inactivity
 Time to wait before stopping instances:
 minutes

Keep one instance started at all times

Keep multiple instances started at all times
 Number of instances:

Maximum number of cluster instances

Limit the number of instances that can start
 Number of instances:

Do not limit the number of instances that can start



You may have the requirement that a minimum number of servers in a dynamic cluster be running at all times; for instance, you may want to ensure that at least three server instances are running at all times. You may also need to limit the maximum number of server instances that can be active; for instance, you may want to specify that you never want more than five instances of a dynamic cluster running, even though many more instances are defined. The main configuration panel for a dynamic cluster gives you control over the minimum and maximum number of instances of the dynamic cluster that can be simultaneously active. The default minimum value is one server, and the default maximum is the number of member servers defined.

You can also specify that the application placement controller can stop all instances of a given dynamic cluster if it is not accessed for a defined period. This can be useful for rarely used applications for which users are willing to wait for application startup, allowing resources to be used for other, more active dynamic clusters.

Dynamic cluster vertical stacking

- Vertical stacking
 - ▶ Can host multiple instances on single node
 - ▶ Appropriate if application is not processor or memory bound

Vertical stacking of instances on node

Allow more than one instance to start on the same node

Number of instances:

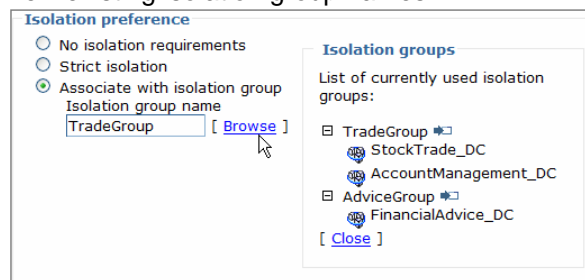


Under normal circumstances WebSphere Extended Deployment assumes that the only resource bottleneck that an application might have is either in the processor or in memory. If you have an application with an internal bottleneck, it may be possible for a single node to host multiple instances of the dynamic cluster running the application. Use vertical stacking to improve bottleneck conditions in your application. With vertical stacking, multiple application server instances can start on a node. By configuring multiple application server instances, you can use all the power that is available on the nodes when a large workload exists for the application.

Vertical stacking is supported on the z/OS operating system, but the WebSphere Application Server for z/OS multiple servant feature is the preferred alternative. Vertical stacking for application servers on the z/OS operating systems should be considered only for application servers that host applications that are constrained to run in a single application server servant region.

Dynamic cluster isolation

- Dynamic clusters can be isolation from other dynamic clusters
 - ▶ Useful when you want particular nodes to run certain applications in isolation from other applications
 - ▶ No isolation requirements - default setting
 - ▶ Strict isolation
 - ▶ Associate with isolation group
 - ▶ Type in new isolation group names in as needed
 - ▶ [Browse](#) to review existing isolation group names



8

Configuring dynamic operations

© 2007 IBM Corporation

Dynamic cluster isolation specifies the isolation requirements that the Application Placement Controller will apply when determining where an instance of the dynamic cluster should be placed. There are three options when configuring the dynamic cluster isolation requirements: no isolation requirements (which is the default policy setting), strict isolation, or associate with isolation group.

“No isolation requirements” is the default isolation policy and depicts current dynamic cluster isolation behavior. It denotes to the application placement controller that instances of the dynamic cluster can be co-located with any other running process when placed on a node.

“Strict isolation” denotes to the application placement controller that when an instance of the dynamic cluster is placed on a node, it must not be co-located with any other running instances of other dynamic clusters. In other words, it can only be co-located with other vertically stacked instances of itself.

“Associate with isolation group” is a convenient way to specify a collection of one or more dynamic clusters whose running instances can be co-located with each other. It is valid for the application placement controller to place running instances of a dynamic cluster with an isolation policy of “Associate with isolation group ”xxx” on the same node as other running dynamic cluster instances, so long as those running instances are of dynamic clusters that are members of the same isolation group. To create a new isolation group you just type a new isolation group name. If you select the Browse link you will be presented with a list of existing isolation groups.

Dynamic cluster server template

- Server template link enables modifying all cluster members from single location
 - ▶ Looks like configuration page for a single server
 - ▶ Modifying this template affects all servers in the dynamic cluster

The screenshot shows a configuration page for a dynamic cluster. At the top, there are three tabs: 'Configuration', 'Reports', and 'Operations'. Below the tabs, the page is divided into two main sections: 'General Properties' and 'Additional Properties'.

General Properties:

- * Name: AccountManagement_DC
- * Type: WebSphere application server
- Number of running instances: 0
- Operational mode: Automatic

Additional Properties:

- Dynamic cluster members
- Dynamic workload management (DWM)
- Server template** (highlighted with a red circle and a mouse cursor)
- Custom Properties

Each node in the bounding node group has one or more servers configured as instances of the Dynamic Cluster. These server instances are ready to be started dynamically when needed. These server instances are configured based on a server template that defines the configuration for all of the cluster members. This template is used as a single point of configuration for all members of the Dynamic Cluster.

Clicking on the 'server template' link for a dynamic cluster displays a page that looks very similar to the configuration page for a single application server. This page, however, affects the properties of every member of the dynamic cluster. This differs from static cluster members, which must be configured individually after they have been created.

The template is not available if the dynamic cluster is created from a group of existing servers.

Configuring on demand routers

- Create, modify, and manage on demand routers under Servers → On Demand Routers
- Click on the name of the on demand router to configure it just like a server
 - ▶ Manage server attributes like ports, thread pools, cache rules from this panel
 - ▶ Service policies and routing policies are not configured here

Select	Name	Node	Version	Protocol	Status
<input type="checkbox"/>	odr	hostANode01	ND 6.1.0.7 WXDCG 6.1.0.0 WXDDG 6.1.0.0 WXDOP 6.1.0.0 XD 6.1.0.0	HTTP, SIP	

Total 1

On demand routers can be created and configured in the administrative console under the 'Servers' menu item and can be placed on any managed node with WebSphere Extended Deployment Operations Optimization installed. While properties of the on demand router, such as thread pool sizes or cache rules, can be configured in this panel, service policies, work classes, and routing policies are not configured here. Service policies are created under the 'operational policies' menu item, while work classes and routing policies are attributes of individual applications. A wsadmin script, called 'createODR.jacl' is also provided for creating an on demand router from the command line.

Section

Configuring operational policies



This section covers configuring operational policies, which define the work types and goal levels used to drive a dynamic operations environment.

Service policies

- Service policies define a level of importance and a response time goal
- Each service policy contains one or more transaction classes
 - ▶ Transaction classes enable work classes to be mapped to the service policy
 - ▶ Also enables finer-grained monitoring by application, server, or work class
 - ▶ A default transaction class is created for each service policy
 - This is sufficient unless you need finer-grained monitoring or reporting



A service policy defines a level of importance and a response time goal, and allows you to describe how requests into your environment should be treated. In addition to defining goals, each service policy contains one or more transaction classes. As you will see later, a work class is a set of rules used for mapping incoming requests to transaction classes. Therefore incoming requests that match a particular rule are associated with a transaction class. This in turn is contained by a service policy that defines the goals and importance for that request. Each service policy contains a single transaction class by default. However, you can create multiple transaction classes within a service policy, enabling you to differentiate, for monitoring or reporting purposes, between requests that you want to handle with the same class of service.

Creating a service policy

- Create service policies in the Administrative Console under Operational Policies → Service policies

The screenshot displays the 'Define service policy general properties' dialog box in the IBM Administrative Console. On the left, a sidebar lists four steps: Step 1: Define service policy general properties (highlighted with a blue arrow), Step 2: Define service policy goal properties, Step 3: Define service policy memberships, and Step 4: Confirm service policy creation. The main area contains the following fields:

- Name:** Platinum_SP
- Description:** (Empty text area)
- Goal Type:** A dropdown menu with the following options: Average Response Time (selected), Average Response Time, Percentile Response Time, Discretionary, and Completion Time.

At the bottom of the dialog are 'Next' and 'Cancel' buttons. The footer of the console shows 'Configuring dynamic operations' and '© 2007 IBM Corporation'.

Service policies are created in the administrative console under the 'operational policies' menu.

A discretionary goal indicates work which is not high priority and can therefore be scheduled by the on demand router based upon the existing workload in the queue. Any request that is not associated with a specific service policy will default to discretionary for goal type.

The average response time goal type is used for high priority work.

The percentile response time specifies that a given percentage of requests must be handled within the required response time goal.

Completion time is only applicable to compute grid jobs, and specifies the maximum amount of time in minutes that is acceptable for a job to complete.

To view a demonstration of creating a service policy, watch the "Create service policy" demonstration.

Work classes

- Work classes define a certain type of work and associate it with a transaction class
- Requests matching the work class rules will be classified into the service policy that contains that transaction class
- Each work class contains a default policy and an optional list of rules
- If no rules match, work will be classified to a default transaction class



A work class is a set of rules that allows you to differentiate between incoming types of work. Each rule created within a work class has an associated transaction class. Work that matches a rule will be treated according to the service policy that contains that transaction class. Each work class also has a default transaction class, which defines how work that does not match any rules will be treated.

Work classes

- Work classes group work in (protocol, policy) pairs
 - ▶ HTTP, SOAP, IIOP or JMS protocols
 - ▶ Routing policies or service policies
 - ▶ Further specialized by rules
 - Classification
 - Routing
- Rule builder panels provide an easy way to create rules
 - ▶ Rules can also be entered manually



Separate work classes are created for HTTP, SOAP, IIOP, and JMS protocols. A work class can be associated with service policies which specify required response time goals, or routing policies which specify how requests should be routed. Routing policies can be used to route HTTP and SOAP requests between multiple concurrently activated editions of an application, or to selectively reject requests from some clients.

The administrative console includes a rule builder for building complex rules within a work class.

Creating work classes

- Work classes are defined under the Service Policies tab of each application

Then apply the following classification rules

Select	Order	Classification Rule	Build Rule
<input type="checkbox"/>	1	If "uid = 'joe' and protocol = 'HTTPS'" Then classify to transaction class AccountManagement_TC (Silver_SP)	Rule Builder

If no classification rules apply, then classify to this transaction class

Select transaction class:
Default_TC (Default_SP)

Work classes are created as attributes of individual applications. To create a work class, choose an installed application from the 'enterprise applications' menu item, and then click the 'service policies' tab from the application's configuration page, where you can create rules like the one shown here. Rules can define Boolean combinations of classification operands, and each rule specifies a transaction class to which the request will be classified if it matches. If you do not know the exact syntax of the rule you want to create, the 'rule builder' button allows you to create rules using a menu-driven wizard.

Request classification

- Requests can be classified by a number of variables:
 - ▶ Virtual host or Uniform Resource Identifier (URI)
 - ▶ HTTP headers, query parameters, and cookies
 - ▶ Web service and operation name
 - ▶ Client or server IP address, port, and host names
 - ▶ Time
 - ▶ User or group ID



Work class rules can match requests by a combination of several variables, including virtual host or URI, client or server IP address, time of day, HTTP headers, query parameters, and cookies.

Sample classification operands

cookie\$_MyCookieName_ IS NOT NULL

uid\$ibm LIKE 'XDuser%' OR roles\$ibm = 'administrator'

queryparm\$timezone = 'EDT'

clientip4 LIKE '10.12.18%.%' OR clientip6 = '1:2:3:4:5:6:7:8'

HTTPMethod = 'POST' AND protocol = 'HTTPS' AND
port IN (9080,9090,9091)

time\$11:40\$13:30

clienthost LIKE '%.ibm.com' OR operation\$calculate = 'finances'



This slide shows some example rules. These rules can be used to classify a request, or to determine how a request should be routed.

Using the rule builder

- The rule builder provides a set of menus to help you create routing policies and learn the correct syntax
- Can enter rules directly

StockTrade_WC

If HTTP request matches

HTTP patterns:

/StockTrade/IOBound (StockTrade.war)
/StockTrade/memBound (StockTrade.war)
/StockTrade/CPUBound (StockTrade.war)
/StockTrade/sleepBound (StockTrade.war)
/StockTrade/CpuAndSleepBound (StockTrade.war)
.....

Edit HTTP Patterns

Then apply the following classification rules

Add Rule Delete Rule Move Up Move Down

Select	Order	Classification rule
<input type="checkbox"/>	1	<p>Edit rule</p> <p>[Build subexpression]</p> <p>If</p> <p>uid = 'joe' and protocol = 'HTTPS'</p> <p>Then classify to transaction class</p> <p>Default_TC (Default_SP)</p> <p>Validate Rule Cancel</p>

If no classification rules apply, then classify to this transaction class

Select transaction class

StockTrade_TC (Platinum_SP)

19

Configuring dynamic operations

© 2007 IBM Corporation

The rule builder can be used to create a new classification rule for your work class. See the rule builder demonstration to view how the rule builder can be used to create classification rules expressions.

Section

Configuring autonomic managers



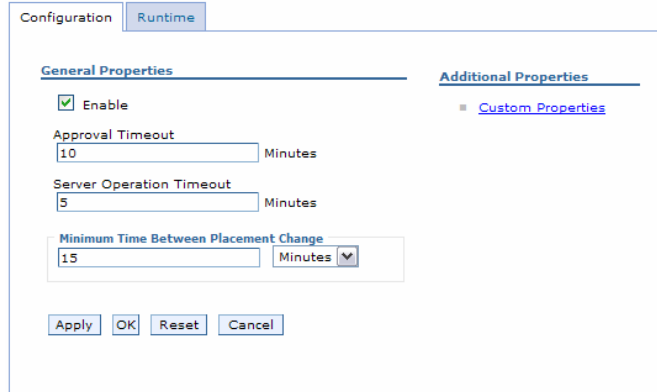
This section will cover the primary autonomic managers that effect dynamic operations and their administrative configuration parameters.

Configure application placement controller

- Operational Policies → Autonomic Managers → Application Placement Controller

Application Placement Controller

Use this page to configure the application placement controller. The application placement controller manages dynamic clusters in supervised and automatic mode.



21

Configuring dynamic operations

© 2007 IBM Corporation

The application placement controller has its own configuration page under the 'operational policies' menu where you can modify its behavior.

The 'approval timeout' defines how long notification tasks live before expiration while running in supervised mode.

The 'server operation timeout' is the amount of time that the placement controller should wait for a server to start or stop before considering the action a failure.

'Minimum time between placement change' is the value that defines how long to wait after performing a set of placement actions until performing the next set of changes.

Configure autonomic request flow manager

- Operational Policies → Autonomic Managers → Autonomic Request Flow Manager

General Properties

Aggregation period
+ 5 Seconds

Control cycle length minimum
+ 59 Seconds

Smoothing window
12 # of aggregation periods

Maximum queue length
1000 requests

CPU overload protection: Maximum percentage of middleware node CPU to use:
90 %

Memory overload protection: Maximum percentage of the WebSphere Application Server maximum heap size to use:
95 %

Request rejection policy for HTTP, SOAP and SIP requests that are associated with a performance goal when an overload condition is detected. Discretionary work is assumed to have a response time threshold of 60 seconds:

- Immediately reject the request if queuing the request would cause it to breach its service policy goal
- Reject the request when queuing the request would cause it to breach its response time threshold value by more than the following percentage 400 %
- Allow the request to remain in the queue even if it means a service policy goal breach.

22

Configuring dynamic operations

© 2007 IBM Corporation

The autonomic request flow manager also has its own configuration page under the 'operational policies' menu. The variables in this panel will be covered in the following slides.

Configure autonomic request flow manager

- Aggregation period
 - ▶ Metrics collection period for operations and charting
- Control cycle length minimum
 - ▶ How often should manager operate

General Properties	
Aggregation period	
* 5	Seconds
Control cycle length minimum	
* 59	Seconds

The autonomic request flow manager gateways collects metrics for each HTTP request. These metric samples must be aggregated periodically for routing determination, server placement, and custom charting. This period is specified by “Aggregation period.”

“Control cycle length minimum” defines how often the autonomic request flow manager controller is activated. When the controller is activated, it evaluates all of its inputs and, if necessary, produces new control settings. The activation process for an autonomic request flow manager controller initiates when new statistics will come from one of its gateways, and if the elapsed time since the previous activation is greater than or equal to the control cycle minimum length, or if the controller has never activated before.

Configure autonomic request flow manager

- Smoothing window
 - ▶ Maintains rolling average over multiple sample periods
 - ▶ Provides protection from transient spikes
- Maximum queue length
 - ▶ Maximum requests per service class and deployment target

* Smoothing window	<input type="text" value="12"/>	# of aggregation periods
* Maximum queue length	<input type="text" value="1000"/>	requests



The autonomic request flow manager controller of any gateway maintains a rolling average of the last few statistic reports from that gateway. The “Smoothing window” controls the number of reports that are combined. A low smoothing window setting makes the controller more sensitive and supports quicker reaction. However, a low setting also increases sensitivity to noise or anomalies in the data.

The autonomic request flow manager maintains a separate queue for each combination of on demand routers, node groups, service classes, and deployment targets. When a request arrives and the queue is full, the request is rejected. “Maximum queue length” bounds the length of each autonomic request flow manager queue to a maximum number of requests that are possibly held in queue. A lower parameter in this field increases the possibility that a request is rejected due to short-term traffic bursts, while a higher parameter allows requests to linger longer in the queues. Queued requests consume memory.

Configure autonomic request flow manager

- Overload protection
 - Processor
 - Memory
- Request rejection policy

* CPU overload protection: Maximum percentage of middleware node CPU to use:
 %

* Memory overload protection: Maximum percentage of the WebSphere Application Server maximum heap size to use:
 %

Request rejection policy for HTTP, SOAP and SIP requests that are associated with a performance goal when an overload condition is detected. Discretionary work is assumed to have a response time threshold of 60 seconds:

Immediately reject the request if queuing the request would cause it to breach its service policy goal

Reject the request when queuing the request would cause it to breach its response time threshold value by more than the following percentage
 %

Allow the request to remain in the queue even if it means a service policy goal breach.

The autonomic request flow manager provides overload protection in addition to its prioritization capabilities. An autonomic request flow manager queues requests in its gateways to avoid overloading the application servers. In version 6.1, load is determined in terms of processor utilization on the tier of application servers immediately behind the on demand router. “Maximum CPU utilization” signals to the autonomic request flow manager how heavily to load these servers. During severe peak conditions, this utilization limit might be exceeded temporarily.

Similarly, “Memory overload protection” specifies the maximum percentage of the heap size to be used for each application server.

You can also specify the behavior for HTTP, SOAP and SIP requests that are associated with a performance goal when an overload condition is detected.

“Request rejection policy” specifies whether and at what point requests should be rejected immediately rather than missing the required performance goal.

Summary

- Configuring dynamic operations requires both creating runtime resources and configuring operational policies
 - ▶ Runtime resources include dynamic clusters, and on-demand routers
 - ▶ Operational policies require service policies, transaction classes, and work classes
 - Work matching a rule in a work class is classified to a transaction class, which associates it with a service policy



In summary, configuring dynamic operations requires creating runtime resources, such as dynamic clusters, and on demand routers, and creating operational policies. Service policies define classes of importance and response time goals, and work is mapped to those policies by work class rules that associate the work with transaction classes.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_XD61_Configuring_Dynamic_Operations



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

