Chapter 32

WebSphere Application Server V6.1 Extended Deployment: Overview and Architecture

The WebSphere Application Server Extended Deployment (WAS XD) package provides many extensions to existing functionality available in the Network Deployment (ND) package in the areas of:

a. Dynamic Operations: Provide Infrastructure Optimization, Intelligent (or Dynamic) Workload Management and Virtualization. This function helps corporations to use the resources of under utilized servers by consolidating the application server environments (clusters and cells).

b. Extended Manageability: Provide Automatic Sense and Response Management. This function allows administration and operations teams to manage and monitor the environment easier and faster.

c. In-Memory Data Caching (or in-memory database)**:** Provide linear scalability, high throughput and improved response times by caching data in memory.

d. Batch Processing: in addition to on-line transaction processing (OLTP) provided in the ND package.

WebSphere Application Server V6.1 XD is available in three individual components. Customers can buy as an integrated package (all 3) or a combination of one or more individual components. The three components are:

1. **Operations and Optimization:** This component comes with all the features available in the Dynamic Operations and Extended Manageability functions described above.
2. **Data Grid:** This component comes with the features available in the "In-Memory Database" function described earlier.
3. **Compute Grid:** This component comes with the features available in Batch Processing feature.

The XD package is an add-on to the ND package (code that runs on top of ND). This means you first need to install and configure ND before you install XD. When you upgrade your environment from ND to XD that means you are moving from standard High Availability (HA) to **Continuously** High Availability (CHA), conventional Workload Management (WLM) to **Intelligent** Workload Management (iWLM), Manual Health and Operational Management to **Automatic** Sense and Response Management, Online Transaction Processing (OLTP) to **both** OLTP **and Batch** processing.

Other leading-edge products like WebSphere Process Server, WebSphere Enterprise Service Bus, WebSphere Commerce Server and WebSphere Portal Server* run on top of a WebSphere Application Server environment, WAS-XD for these layered products is already supported (*being tested in case of Portal Server V6.x).

We use simple scenarios in our discussion to concentrate on the basic features of XD. Good knowledge (and work experience) of WAS ND V6.x is desirable to understand the XD concepts we discuss in this chapter. Before we try to explain the features of WAS-XD let us study the sample ND topologies shown in the diagram (Figure 32-1) below and see some of the limitations we encounter in the scenario that is based on a conventional J2EE platform and how WAS-XD can help us eliminate those limitations using the technology that is beyond J2EE.

ND Topology

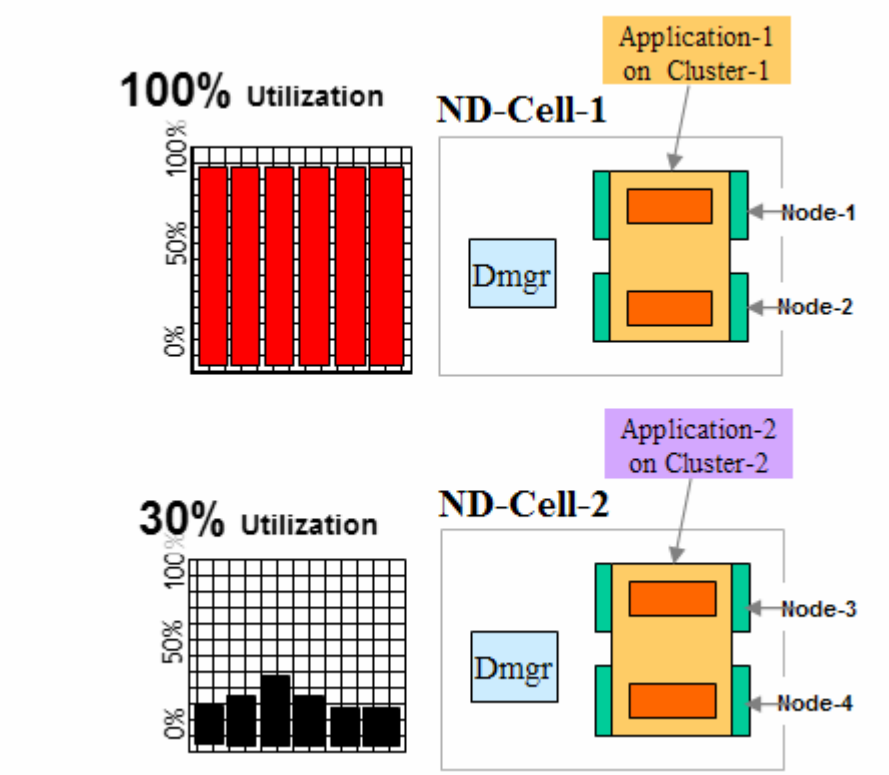

*Figure 32-1: ND environment with unique application on each **static** cluster*

In this sample ND topology we have two clusters (each in a separate cell) each hosting a unique application (App1 on Cluster-1 and App2 on Cluster-2) as shown in the diagram Fig 32-1. Assume that the first application (App1) is not meeting the Service Level Agreement (SLA) due to high CPU, memory utilization and other resource contention during certain periods of the day due to sudden increases in load. Further assume the second application (App2) is well within its SLA targets with low CPU and memory utilization. An ND environment does not provide the resource sharing flexibility needed to dynamically expand or contract the resources available for running a given application. We'd like to be able to add nodes and cluster members for use by App1 when it is getting close to not meeting its SLA. The nodes allocated to App2 are underutilized and potentially available for running App1. But ND does not support a function that allows a

cluster to grow or shrink automatically based on load. We call the clusters created in ND static clusters.

Another limitation with ND is the inability to prioritize incoming Web requests depending on URI. For example, let us assume that serving a URI with '/inv' is more important than serving a '/hr' request. ND has no function to classify the requests depending on importance of the URI. In this case we'd like to handle /inv requests before /hr if these requests come at the same time and there are resource constraints. We'd like a feature that allows a priority to be assigned to requests. We'd like high priority requests to take precedence in processing queues under resource constrained conditions.

<begin note>
This document is an overview of WAS-XD for architects, administrators with no prior knowledge of WAS-XD. To keep things simple; we show just two applications and four application server nodes. However, WAS-XD is intended for environments with tens of mission critical applications using tens (or hundreds) of application servers.
<end note>

In the ND environment, you cannot deploy and manage multiple editions of an application (multiple versions of the same application having the same context root). In addition, ND does not provide the capability to relatively easily roll out a new edition (especially with validation before activating) of an application or roll back to a previous edition without a service interruption.. An application edition is a unique build version of an application or same the build version with different deployment bindings or both.

In addition ND does not provide integrated health monitoring (even though ND provides some performance metrics through the Performance Monitoring Infrastructure (PMI), there will be noticeable performance impact depending on the level of monitoring defined. That is why PMI is usually enabled to solve performance related issues and disabled once the problem is resolved). Health monitoring tells you how each cluster member is doing (memory leaks or high CPU/memory/resource utilization for example), and how well applications are performing. (Are they meeting SLA's ?, for example.) In addition to being able to view the health condition of your applications and application servers, you also want to notify an administrator or take corrective action before a failure condition occurs. ND **does not** directly support **alarming and notification**.

ND does not directly support a highly available Deployment Manager (DM). (You need to use operating system or hardware level high availability software such as HACMP.) A highly available DM is a must if you want to monitor health and take appropriate steps to correct runtime environment problems through the Deployment Manager's administrative console.

ND also does not support **automatic** incremental configuration backup (just like database incremental backup) as you make changes to the configuration. This makes it more difficult for an administrator to revert (from faulty configuration) to the previous configuration if required.

Installing product binaries and updates (refresh packs, fix packs and interim fixes) for each node is a tedious process in an ND environment. Even though ND V6.x supports sharing product binaries from a network file system (highly available NFS for example) on each WebSphere node, it is not popular except in Z-Series environment because product binaries accessed over a network from each node will reduce performance.[1]

The limitations in ND we've discussed above are addressed in the XD package. Remember that we are only trying to provide a general overview of XD in this online document for the administrators and architects who are already familiar with the ND environment. Complete coverage of the XD package deserves a full book itself. We recommend you review the Extended Deployment InfoCenter for more information on this product at http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1/index.jsp.

XD Topology

After knowing some of the limitations of ND (which is built on conventional J2EE technology), we will see how XD can better utilize existing processor resources to meet SLAs for the applications running in the environment. In addition we will see how XD provides a stronger environment for, operating, managing and monitoring applications and application servers. The diagram shown in Figure 32-2 depicts how two or more ND cells (or multiple static clusters in a single cell) can be combined (or virtualized) to share resources among cluster members to meet SLAs for resource constrained applications.

We combined ("pooled") nodes from both of the ND cells and created an XD node group. Each application (App1 and App2) is deployed to a separate dynamic cluster. (The number of cluster members will expand or contract as needed to meet SLAs.) In this example App1 is deployed on Dynamic Cluster-1 and App-2 on Dynamic Cluster-2. We describe the elements of the XD package take steps to share resources in order to meet SLAs for both of the applications. Dynamic cluster-1 is expanded to include two more JVMs on node 3 and 4 to meet its SLAs. Meanwhile, the XD configuration is able to keep the overall utilization to 55%. As overall utilization in virtualized XD environment is within limits (55% in our example and still meeting SLAs for both the applications), organizations can try deploying a third application (App3 for example) without investing extra money on procuring, installing and configuring new hardware. Another way of architecting this XD environment is by deploying all the applications (App1 and App2 in this case) on a single dynamic cluster that spans across four nodes and let XD decide where (which JVMs) to process the application's request depending on the service policies defined for each application.

[1] You can download a document (chapter 28) on setting up shared product binaries from www.WebSphereMentor.com.

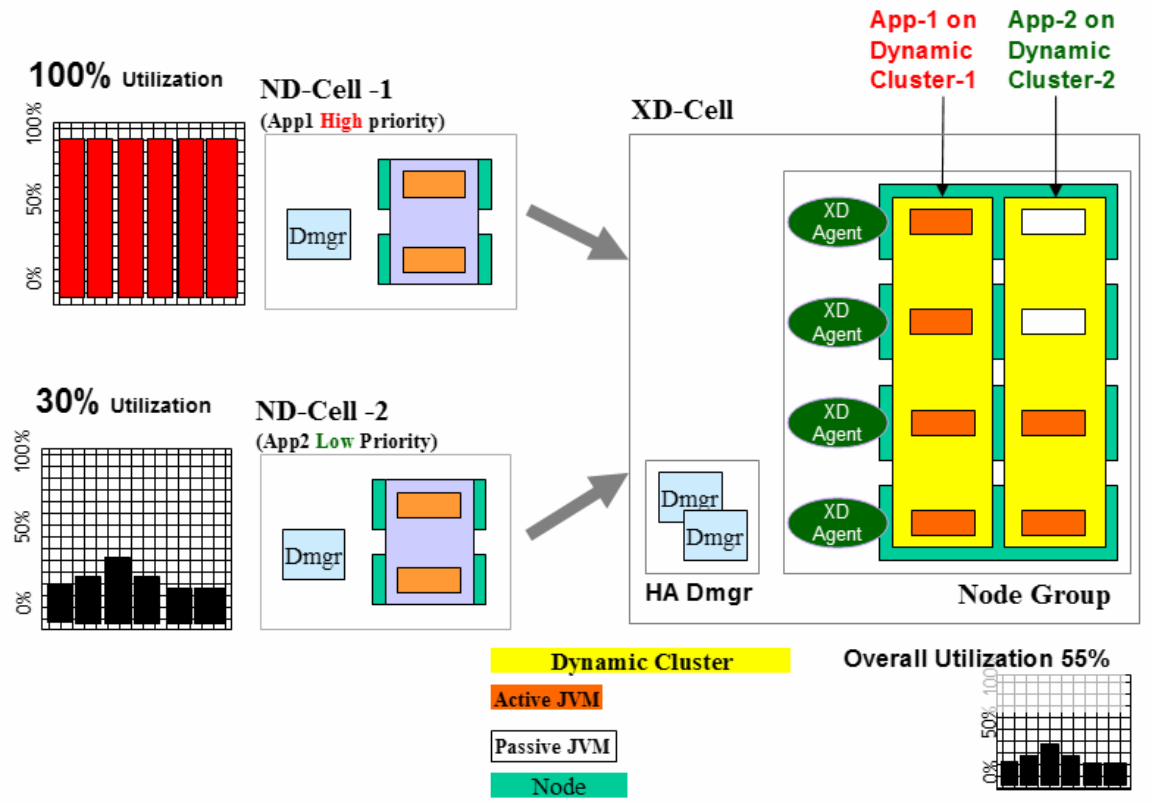

Figure 32-2: Example of converting WAS-ND static cluster environment to WAS-XD environment with Dynamic clusters

<begin note>
In WAS-XD V6.1 dynamic clusters are not confined to a fixed number of pre-determined nodes. Dynamic clusters may span nodes that satisfy a membership policy expression for example, hostnames starting with 'was-host'. The screen shots show Node Group to make things easier, but a dynamic cluster is not confined to a node group as shown in Fig 32-2.
<end note>

What happens if there is sudden increase in traffic for App-2 too? In such a situation, the traffic is queued and prioritized depending on the service policies you gave to an application's URI. You define and map service policies after deploying an application. The On Demand Router (ODR) component takes care of queuing and prioritizing requests as shown in Fig 32-3. The ODR is a type of server available in WAS-XD that was built on top of the Proxy server available starting with version 6.0.2 ND.[2] ODR is an extension of the ODC (On Demand Configuration) features available in Proxy server to cater to On Demand Environments using the features like Request Classification, Flow Control/Queuing, Prioritization, Dynamic Workload Management and cross-cell routing. The ODR intelligently routes HTTP and SIP requests to the XD enabled dynamic cluster

[2] Refer to chapter-27 (free download) for more information on Proxy server from www.WebSphereMentor.com.

members. Requests are routed based on the service policies defined in the XD configuration and feedback from the node agents on utilization and "health" of the cluster members.

In Fig 32-2, you can also see that the deployment managers have been configured in highly available mode. There will usually be one primary deployment manager and one standby deployment manager in an XD environment. The standby deployment manager will be activated if the primary is down.

<begin note>
The features of XD are useful in scenarios where different applications have different demand profiles that have peaks at different times. Not every ND topology (or set of ND topologies) is suitable for an XD environment For example, if the peak load for App1 starts at 8AM and extends through 10PM and the peak load for App2 is from 1PM-3PM every weekday, XD cannot work miracles by reallocating constrained nodes. More resources must be made available to run the applications.
<end note>

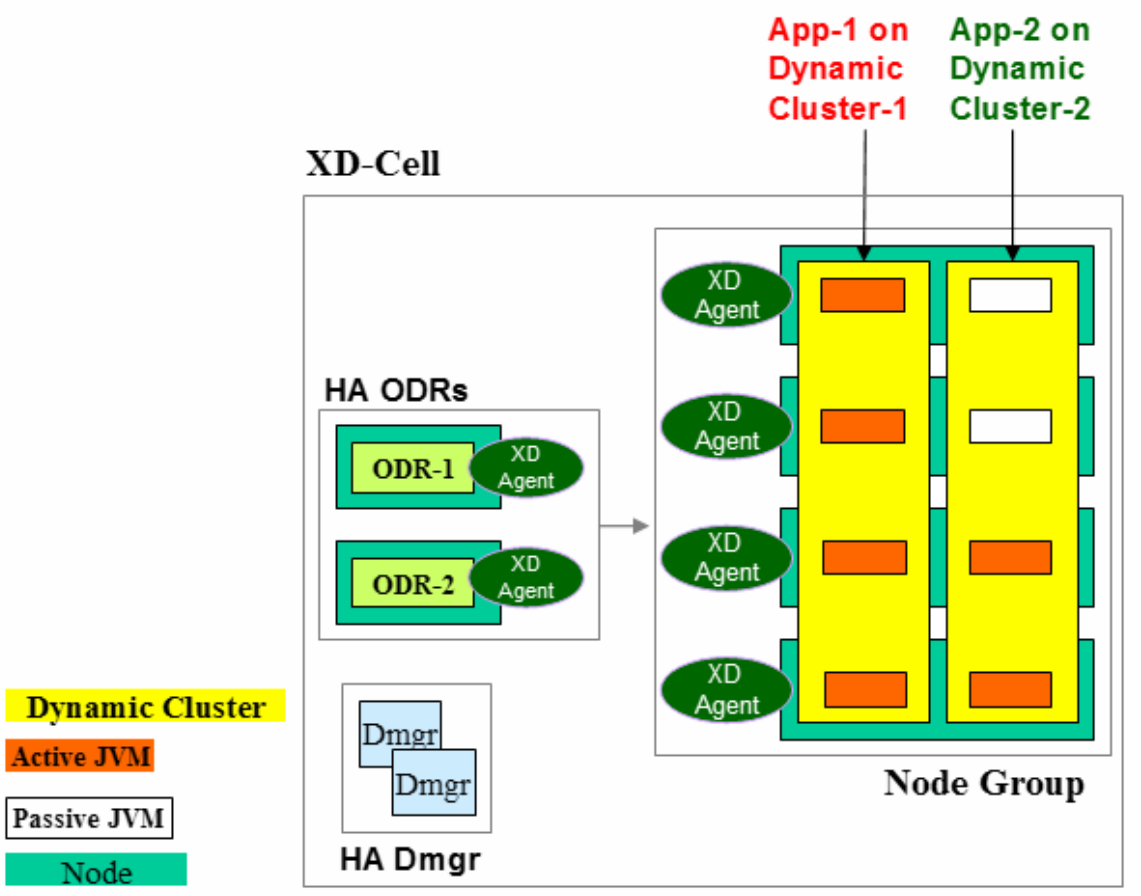

Figure 32-3: XD environment using HA ODRs, HA DMGRs and Dynamic clusters

Fig 32-4 above gives an idea of sample XD topology that is closely related to a typical ND topology. The new XD components are shown on top of ND with different layers

from front-end through the backend server layers (Edge Server, HTTP servers, ODR, deployment managers and dynamic cluster members in highly available mode). Any specific ND or XD architecture depends on the specific requirements of the enterprise or organization, as well as the applications that are going to be deployed. Be aware that this architecture is shown just as an example.

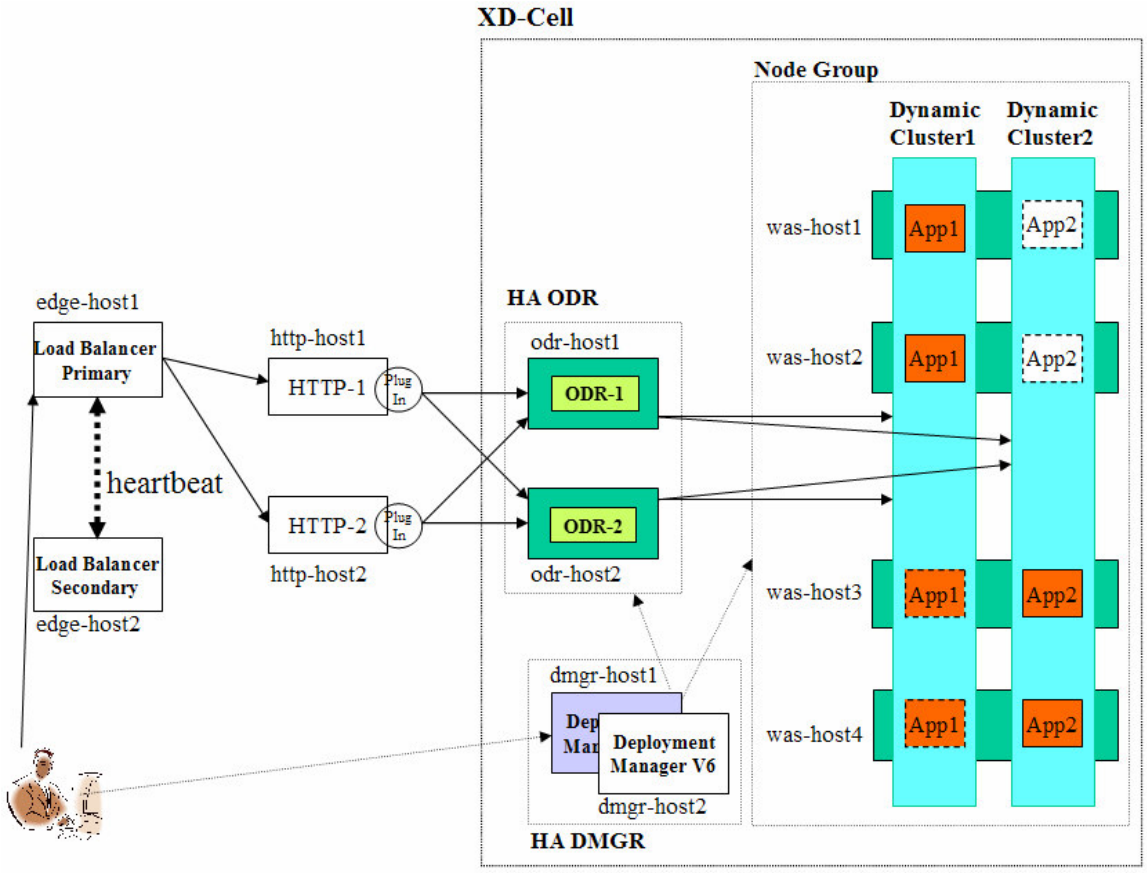

Figure 32-4: WAS-XD topology from front-end thru backend dynamic clusters

So far you have seen some highlights of the dynamic operational features of WAS-XD. Now let us see some important **extended manageability** features. One of them is the **application edition** feature. With XD you can deploy multiple editions of the same application (only one of the editions will be active and serve the user requests (except during transition from old to new edition for a brief period depending on the option you choose). You can **validate** the newly deployed application edition (edition 2 for example) by giving access to only a subset of users (testers for example). Then you can activate the new edition after it has been certified by testers. The old version of the application will process the requests of users, while you validate and activate the new edition. WAS-XD will upgrade to the new edition without interruption of service to existing or new requests. WAS-XD in this case will quiesce a subset of cluster members (while the remaining cluster members serving requests through the old edition), and map the new edition to the quiesced cluster members and then restart those cluster members. Once the new edition starts serving requests through the cluster members, the remaining set of

cluster members will be upgraded using the same process. You will also have old edition(s) of the application kept in the repository, which allows you to rollback to the old edition should the new edition pose problems. Fig 32-5 shows the screenshot of an application edition page that demonstrates the features we explained.

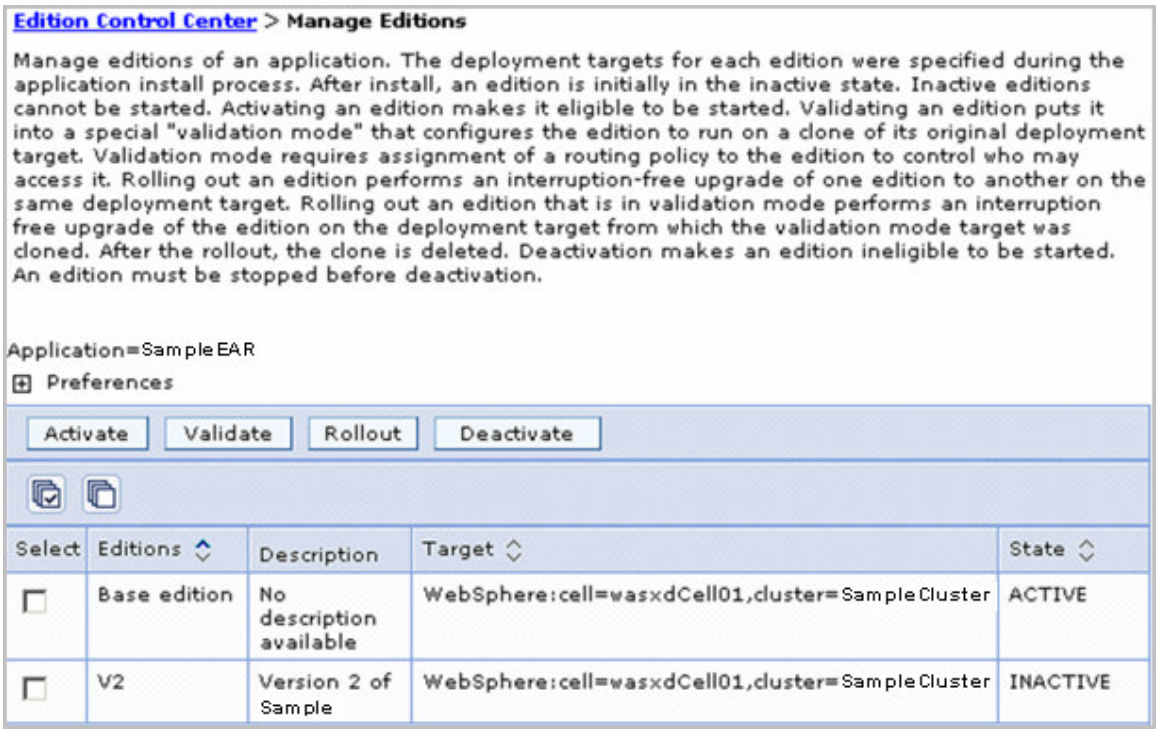

Figure 32-5: Deployment of multiple editions of the same application

Another useful feature is administrator notification. For example XD can send a notification to an administrator by email that describes a task to complete on the admin console. Or you can define a notification to be sent that describes a corrective action to take when a certain event occurs. Suppose you suspect a poorly written application has a memory leak. WAS-XD can detect a memory leak condition by tracking the available memory after a garbage collection (GC) occurs for a certain period time. Fig 32-6 shows the screenshots of 1. Different error conditions you want XD to keep track of (excessive memory usage, request timeout, response time, memory leak etc.), 2. Corrective action you want XD to take if that error condition occurs (taking heap dump and restart server in our example), 3. Email that is sent by WAS-XD to an administrator notifying about the error condition. You can get the MDD4J (Memory Dump Diagnostic for Java) that reads the heap dump and identifies suspicious class types. The IBM Support Assistant is a common platform for launching MDD4J and Thread Analyzer for reading thread dumps.[3]

[3] If you need information on the ISA tool then refer to the chapter on ISA (chapter 30) that is available as a free download from www.WebSphereMentor.com.

Figure 32-6: Health policy configuration and notification of breach condition

The screenshot below (Fig 32-7) shows how WAS-XD can create backup copies of a configuration in order to revert back to that configuration should a problem occur with the existing configuration. This backup and recovery works almost like a database backup and recovery process. WAS-XD will allow you to take a full backup and then it will automatically take a series of delta checkpoints as you make changes to the configuration. You can quickly restore to a stable state from a bad configuration by applying delta checkpoints in the reverse order by selecting the delta checkpoint(s) and clicking on restore.

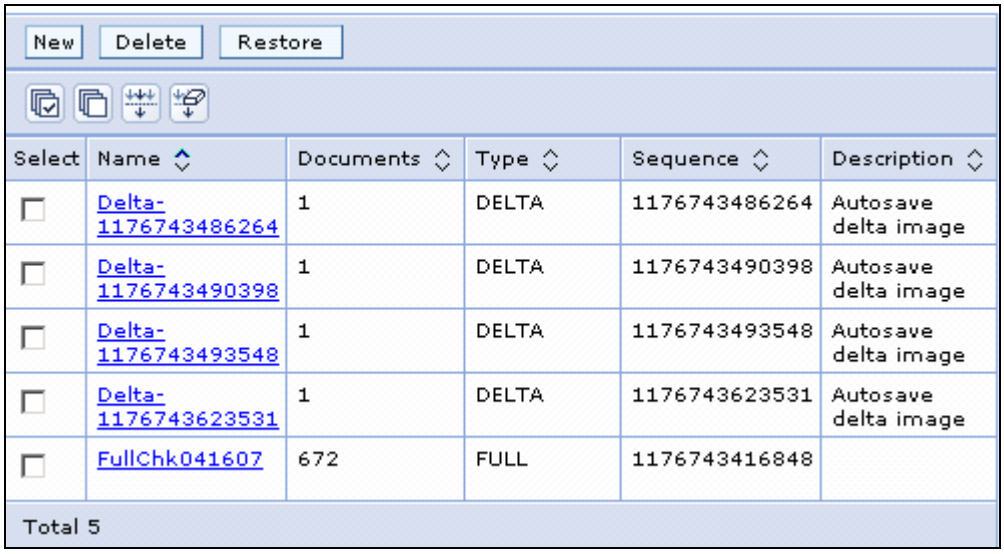

Figure 32-7: Backup and recovery of XD configuration

The screenshot in Fig 32-8 is showing how WAS-XD can help administrators and operators by giving a cell-wide (multi-cell-wide in multi-cell environment) runtime view. XD provides in-depth reporting charts for example collecting and displaying data on CPU, memory utilization of application server nodes/clusters/cells and, response times of the applications. These charts are real-time snapshots of the WAS-XD environment. WAS-XD V6.1 also captures data on applications, resources, users and workload

information to generate reports on utilization and performance statistics (much more than what Tivoli Performance Viewer provides). WAS-XD V6.1 can also generate log files that can be an input to the IBM Tivoli Usage and Accounting Manager (ITUAM) tool to charge back customers based on the resource utilization. For more information on the charge-back topic, refer to Ann Black's blog at *http://www-03.ibm.com/developerworks/blogs/page/insidexd?entry=charge_back_metrics_for_xd.*

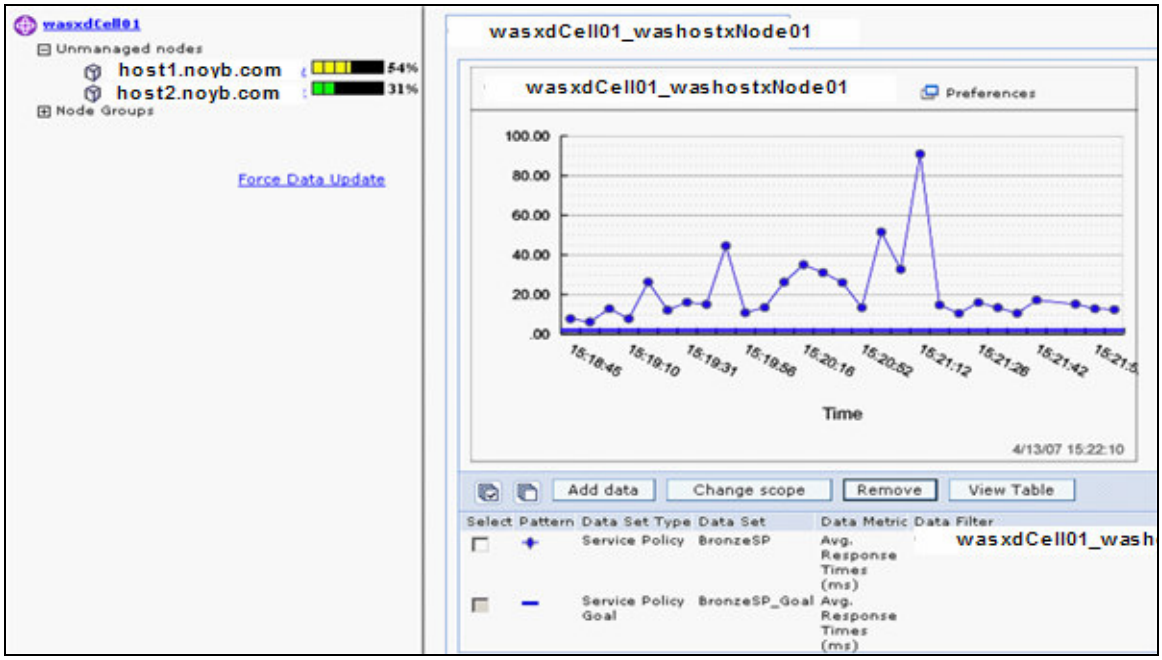

Figure 32-8: Cell wide health condition view

Another great feature introduced in WAS-XD V6.1 is centralized installation and patch management from a single point through deployment manager. You can install XD updates on just the deployment manager and schedule the updates to desired nodes and WAS-XD will automatically update the nodes.

From WAS-XD V6.1 with its enhanced features, you can also use some or most of the XD operation and optimization features (like dynamic clusters, health monitoring, application edition) in Non-WebSphere Application Server environments like PHP, Germino, Tomcat, BEA Web Logic, and JBoss for example. This is possible by installing a lightweight XD's middleware agent on each Non-WebSphere node and then federating that node to the extended deployment environment. The level of XD support on Non-WAS servers varies from product to product and the product version. Refer to WAS XD V6.1 infoCenter for more information.

There are other new features supported in WAS-XD V6.1 that are beyond the scope of this document. I hope this document gave a good jumpstart on the Extended Deployment package to discuss with your colleagues and management to see if it right for your organization. For more information refer to the InfoCenter link provided earlier in this document.