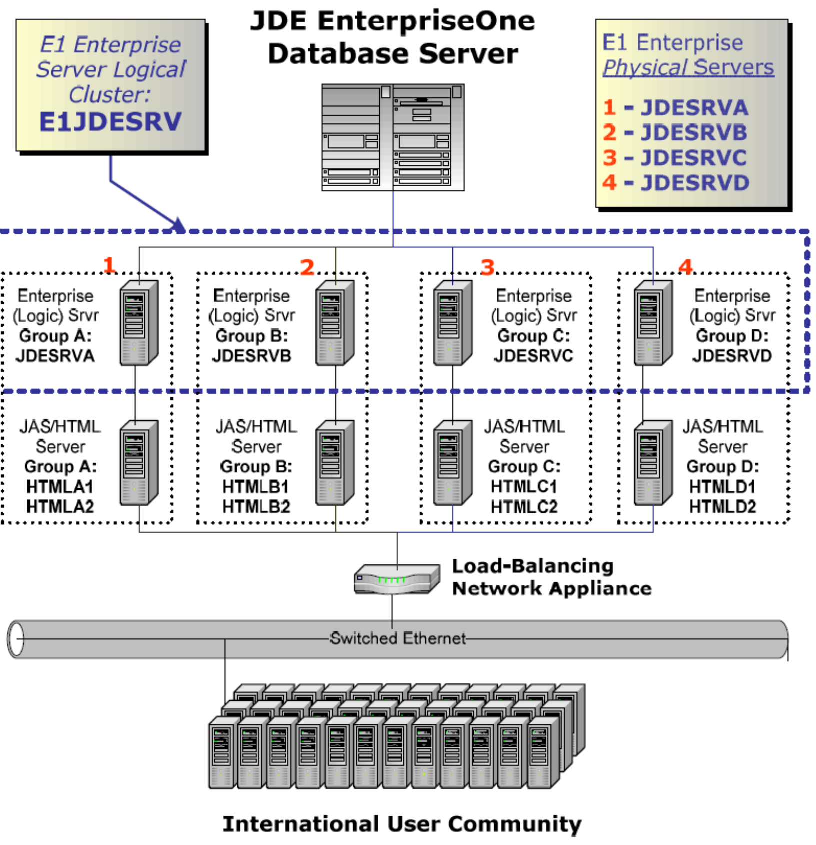
24/7 High Availability Operations

Oracle JDEdwards EnterpriseOne customers whose businesses run around the clock (**24/7**: *twenty four hours a day, seven days a week*) may consider using high availability solutions. Oracle JDEdwards does not explicitly support high availability ‘active-active’ load balancing cluster solutions for the JDEdwards Enterprise Server (application/logic/batch *server*). These types of solutions rely heavily upon the use of specially architected third-party hardware and software products. That said, there are many Oracle customers who run both JDEdwards EnterpriseOne JAS/HTML *and* application servers in a high availability cluster arrangement. In many cases 24/7 customers use a network appliance to NAT the IP address of a logical cluster hostname to individual backend node servers. This document does not discuss the configuration details (e.g. sticky IP) necessary to use or a network appliance for clustering, rather it is intended to convey JDEdwards EnterpriseOne support and maintenance considerations.

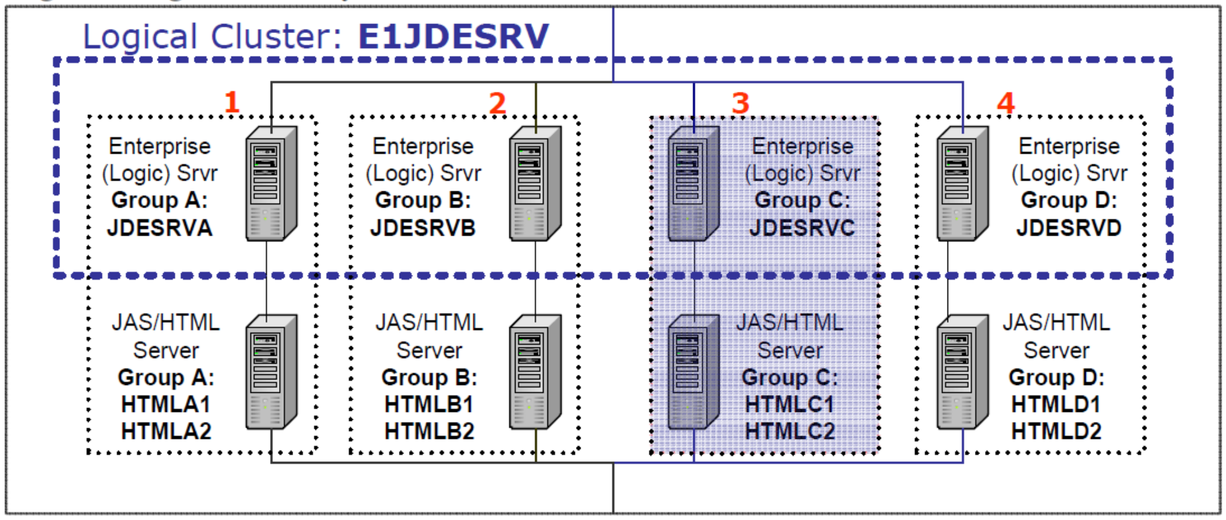
Figure 1 below is an architecture diagram example depicting a high availability cluster. In this example there are four physical server nodes which participate as a single logical cluster server. Associated with each JDEdwards EnterpriseOne Application Server node is a set of HTML Java Virtual Machines (JVMs). These server sets are delineated by design. The importance of pairing a logical server and HTML JVMs into a set will be discussed later in this document. When an enduser request for EnterpriseOne HTML access is sent to the load-balancing network appliance it is intelligently routed to an HTML server which is linked to an application server

Figure 1: High Availability JDEdwards EnterpriseOne Cluster


E1 Enterprise Server Logical Cluster:
E1JDESRV
JDE EnterpriseOne Database Server
E1 Enterprise Physical Servers
1 - JDESRVA
2 - JDESRVB
3 - JDESRVC
4 - JDESRVD
1
2
3
4
Enterprise (Logic) Srvr Group A: JDESRVA
Enterprise (Logic) Srvr Group B: JDESRVB
Enterprise (Logic) Srvr Group C: JDESRVC
Enterprise (Logic) Srvr Group D: JDESRVD
JAS/HTML Server Group A: HTMLA1 HTMLA2
JAS/HTML Server Group B: HTMLB1 HTMLB2
JAS/HTML Server Group C: HTMLC1 HTMLC2
JAS/HTML Server Group D: HTMLD1 HTMLD2
Load-Balancing Network Appliance
Switched Ethernet
International User Community

Oracle JDEdwards EnterpriseOne application release E812 and higher introduced a change to the manner in which the metadata specifications (SPECs) are stored *and* tracked. In prior releases the JDEdwards product SPECs existed both in a relational database form and as a set of 'replicated objects' in TAM SPEC files (single level file storage). With application release E812 and higher the SPECs exist largely in database tables.
They are stored as XML specifications encapsulated within binary large objects. When SPEC updates are 'deployed' there exists a dynamic auto-discovery process. This process ensures that the HTML serialized objects remain in sync with the deployed metadata specifications. Consequently for release E812 and higher, each HTML server associated with a common pathcode and logical server name will find their serialized objects will be updated.
This is true even in cases where each JAS HTML server uses mutually exclusive databases to store their serialized objects. Thus, the auto-discovery process becomes an issue for customers who have become accustomed to removing an Application & JAS/HTML server set one at a time from the 'active' cluster to perform maintenance, e.g. deploy a package. The moment the removed Application & JAS/HTML server set is updated with a deployed package ***ALL*** of the JAS/HTML servers sharing the same pathcode will dynamically auto-discover and update their HTML serialized objects.

Figure 2: High Availability Cluster Server Node Sets


Logical Cluster: E1JDESRV
1
2
3
4
Enterprise (Logic) Srvr Group A: JDESRVA
Enterprise (Logic) Srvr Group B: JDESRVB
Enterprise (Logic) Srvr Group C: JDESRVC
Enterprise (Logic) Srvr Group D: JDESRVD
JAS/HTML Server Group A: HTMLA1 HTMLA2
JAS/HTML Server Group B: HTMLB1 HTMLB2
JAS/HTML Server Group C: HTMLC1 HTMLC2
JAS/HTML Server Group D: HTMLD1 HTMLD2

Referring to **Figure 2** above, consider the following Package Deployment scenario for JDEdwards EnterpriseOne E811_SP1 used by a 24/7 operation. During a maintenance cycle:

1) A server set (Server Group C) will be made ineligible to *new* users.

2) Deploy the new full/update client package to the web generation machine.
The web generation machine's JDBJ.INI file must be updated to point to a serialized objects database location that is not currently in use. Begin the web generation process.

3) Once the '*ineligible*' server set is free of users and all batch activity is held, shutdown the HTML servers. Submit the Package Deployment process to the Logic Server.

4) Update the HTML Server JDBJ.INI files setting the serialized objects location to that used in step 2 above.

5) Review the logs for a successful package deployment and web generation process.

6) Restart the JDEdwards EnterpriseOne Enterprise Server and JAS/HTML servers.

7) Test the server set by connecting to the HTML servers by physical IP address (as opposed to the logical IP address) so as to bypass the load-balancing mechanism.

8) Once the server set is certified, make the server set eligible for use within the cluster.

9) Repeat this process for each server set until all sets have been updated with the new deployed package.

For JDEdwards EnterpriseOne application release E812 and higher, special INI settings are required to accomplish the process above. By setting the JAS.INI `PackageDomainServer` and `PackageDomainPort` values to its associated Application Server, the auto-discovery process restricts its assessment to the scope of the set server value.
Further by updating the JDBJ.INI file to use a distinct set of Serialized Objects for each server set, the XML SPEC package autodiscovery and update process will not interfere with other running HTML servers.

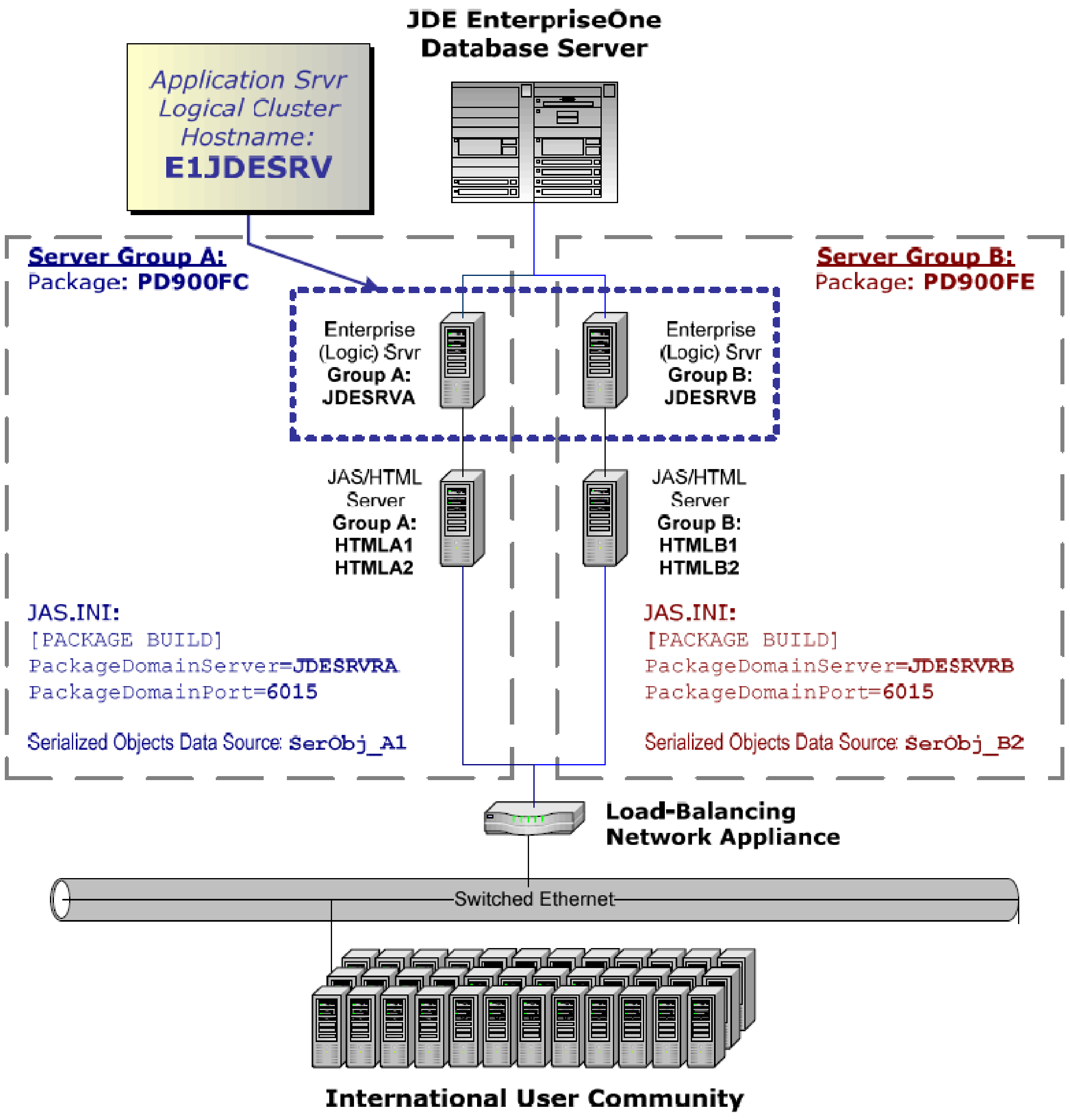
JAS.INI:

```
[PACKAGE BUILD]
;PackageDomainServer=DEFAULT
;PackageDomainPort=0
PackageDomainServer=JDESRVA
PackageDomainPort=6015
```

JDBJ.INI:

```
[JDBj-SPEC DATA SOURCE]
...
name=SerObj_A2
database=
server=JDEE1DB
physicalDatabase=SEROBJA2
```

Figure 3: JDEdwards EnterpriseOne >= E812 Cluster



. . .