Hewlett Packard Enterprise

HPE NonStop Application Server for Java

Enterprise Java on HPE NonStop servers

Powerful platform
HPE NonStop Application Server for Java 1.4 is based on JBoss Application Server version 7.1.2, which is Java Enterprise Edition 6 (JEE 6) full-profile certified.

Introduction

What is NSASJ?

HPE NonStop Application Server for Java (NSASJ) brings the popular open source Java Enterprise Edition 6 (JEE 6) Application Server—namely, the JBoss Application Server (AS)—to HPE NonStop servers. Continuing on the successful legacy of previous releases, NSASJ 1.4 provides standards-based enterprise application development and deployment platforms and tools to its users.

About NSASJ 1.4

NSASJ is a value-added port of JBoss AS release 7.1.2. The product includes the Enterprise JavaBeans (EJB) Container and the Web Container modules of the JBoss AS. NSASJ also includes an implementation of related technologies such as the Java Transaction API (JTA), and it offers the ability to manage the container through a Web-based management console. The JAX-WS Web Services and JAX-RS RESTful Services stacks enable applications deployed in NSASJ to be reached and utilized by clients deployed in diverse platforms across networks.

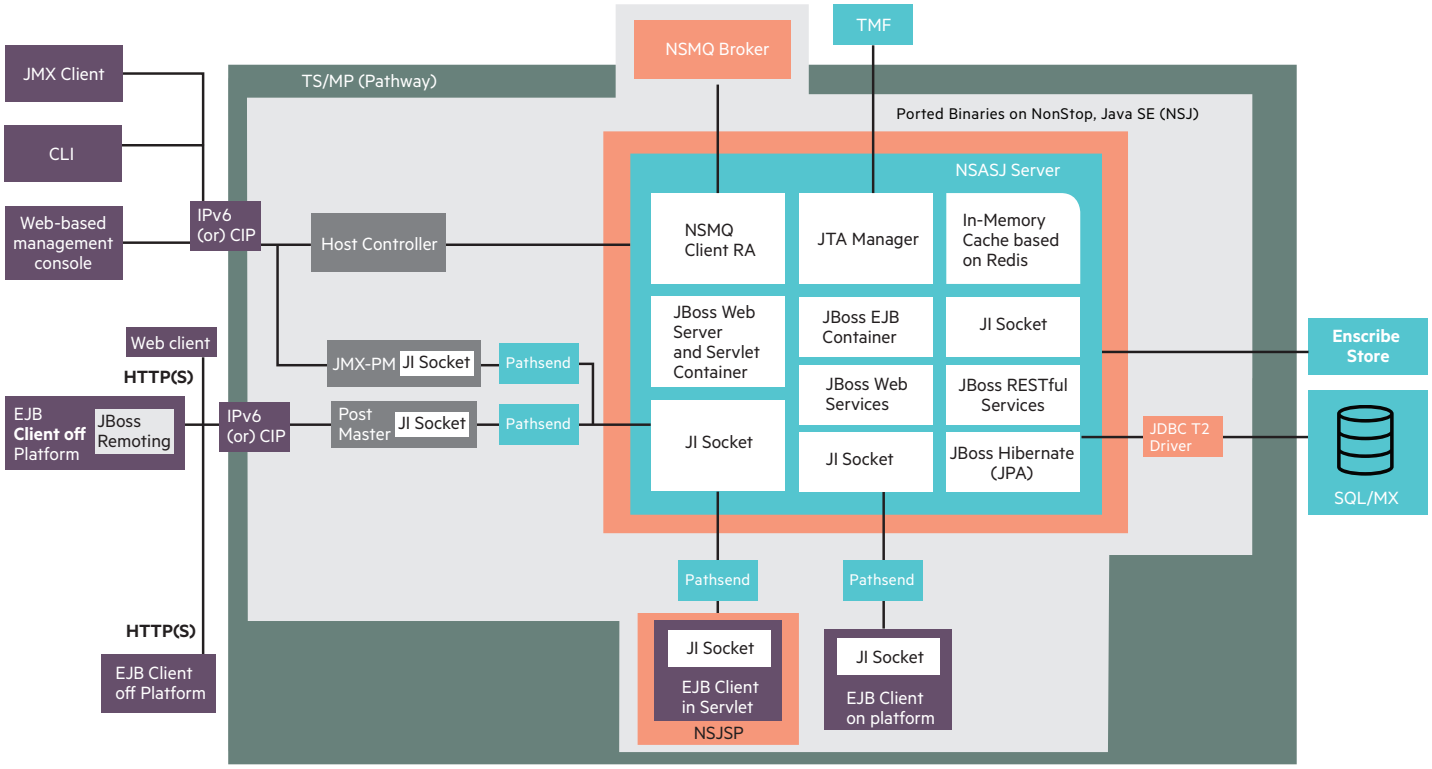
EJB technology enables the creation of reusable components for server-side logic. This technology helps you create complex, backend business logic code in enterprise applications faster, more easily, and in a standardized manner. Doing so enables portability across various server platforms. In short, EJBs extend the write once run anywhere promise of “plain old Java objects” to enterprise-class applications.

NSASJ 1.4 complies with EJB 3.0/3.1 specifications, which are part of the JEE 6 profile. EJB 3.0 also brings the power of Java Annotations, which were standardized in JEE 5.0 specifications. Using annotations reduces the number of configuration settings required for EJB deployment and lifecycle maintenance. EJB 3.x simplifies container-managed persistence by using Java Persistence API (JPA), which is implemented in the JBoss AS using Hibernate.

The concept of Web Containers in JEE has a similar functionality; container helps application developers concentrate on the Web application’s primary function of delivering dynamic content to Web clients. For building and running Web-based applications, the Java EE platform provides a servlet container and a Java runtime environment. The container provides Java servlet APIs and JavaServer Pages (JSP) objects, class libraries, and resources such as HTML and/or XML documents and images. The servlet container also performs runtime resource management, including initialization, invocation, and management of the servlet and JSP lifecycles.

The container also provides deployment descriptors (web.xml file) that contain resource definitions such as Multipurpose Internet Mail Extensions (MIME) types, mapping of requests to servlets, access control, and servlet initialization parameters. The JSP technology allows Web developers and designers to rapidly create and easily maintain information-rich, dynamic webpages for new and existing business applications. NSASJ 1.4 complies with Servlet 3.0 and JSP 2.2 specifications of the JEE.

NSASJ is integrated with NonStop TS/MP and hence it inherits the continuous availability & near-linear scalability that Pathway serverclasses are famous for.



JI—Java Infrastructure
JPA—Java Persistence API
JTA—Java Transaction API
NSASJ—NonStop Application Server for Java
NSMQ—NonStop Message Queue
IMC—In Memory Cache
PM—Post Master

Figure 1: NSASJ 1.4 architecture

What's new in NSASJ 1.4?

This release brings the following new features:

- **Request Retry**—In case of certain failures while handling a client's request, the NSASJ ecosystem will retry the ongoing transaction automatically by itself with no additional effort from the client tor the server-side application
- **In-memory cache** using Redis cache server
- **JMX Management Interface**
- **Silent Installation**
- **nsasj_info_collector.sh**

Key features continuing from NSASJ 1.3

- The JTA functionality of NSASJ has been integrated with the NonStop TMF. Hence all JTA transactions can be part of single, global TMF transaction
- Web Services (JAX-WS) and RESTful Services (JAX-RS)
- Web application container from JBoss AS
- Compliance with Java Servlet 3.0 specifications, JSP 2.2 APIs, and Expression Language (EL) 2.2
- Compliance with EJB 3.0/3.1 specifications of the JEE 6 profile
- Support for Stateless Bean, Stateful Bean, and Message Driven Beans
- Support for both on- and off-platform clients
- Continuous availability and near linear scalability
- Solid data integrity through seamless integration with NonStop Transaction Management Facility (TMF)
- Support for EJB session persistence through JPA and Hibernate
- Integration with NonStop Message Queue (NSMQ)
- Support for 64-bit mode operations
- Management through a CLI and CLI-based GUI, provided by JBoss
- Management of EJB and Web Containers through a Web-based management console

Architectural highlights

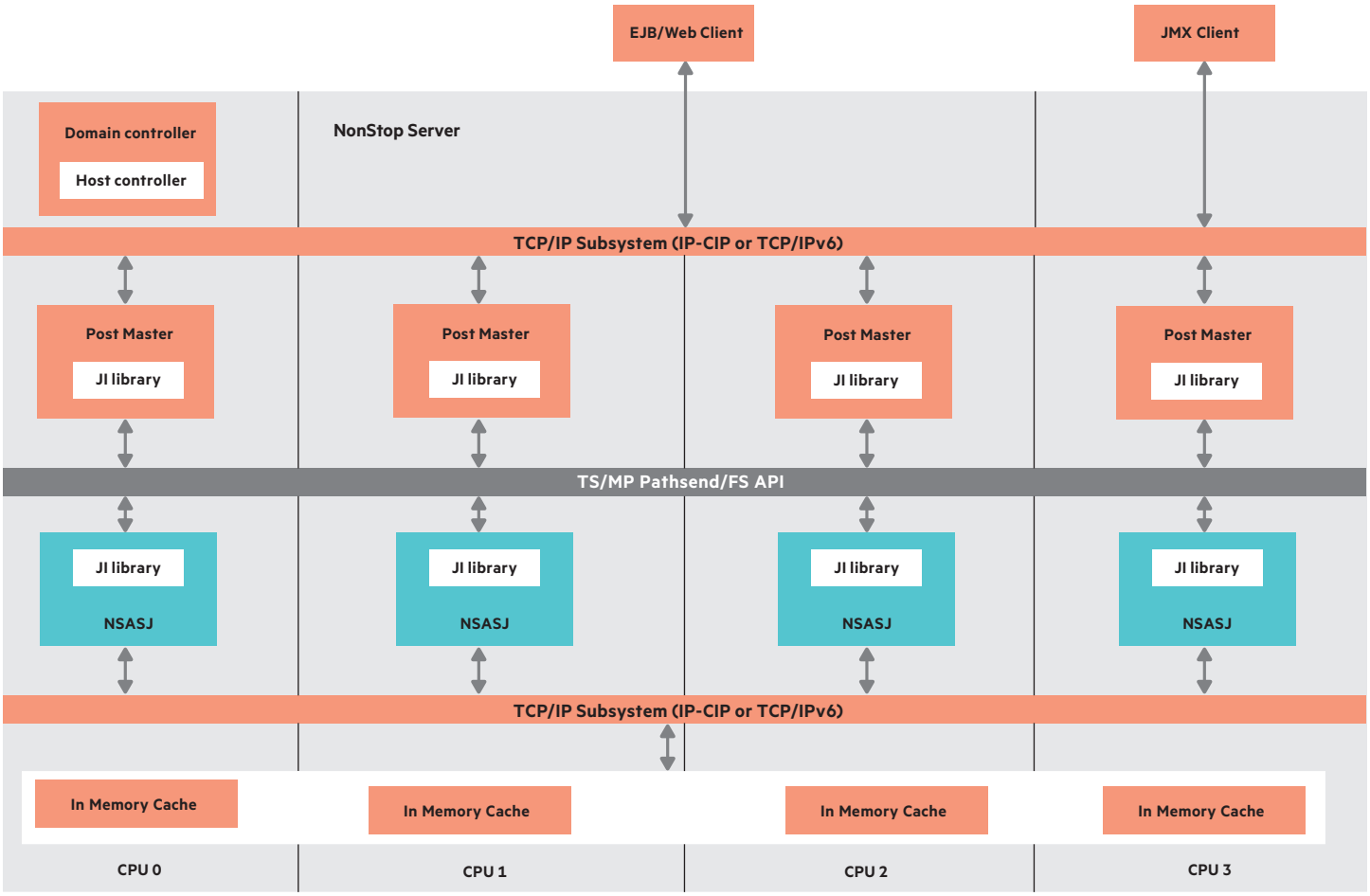


Figure 2: NSASJ Serverclass

Did you know?
EJB 3.x specifications were developed with a special focus on simplicity to enable faster development of enterprise applications.

EJB 3.0/3.1

The built-in capabilities of Enterprise JavaBeans enable developers to concentrate solely on their business logic, rather than worry about platform dependence and related functionalities such as transaction integrity, data persistence, and security. NSASJ supports all three types of beans defined in the specifications—Stateless Session Bean (SLSB), Stateful Session Bean (SFSB), and Message Driven Bean (MDB).

Servlet 3.0

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server, as well as for accessing existing business applications. The servlet technology provides a component-based, platform-independent method for building Web-based applications. Servlets have access to the entire family of Java APIs, including the Java Database Connectivity (JDBC) API. Servlets can also access a library of HTTP-specific calls.

The Servlet 3.0 specifications deliver the following new features:

- Asynchronous request processing
- Web fragments, which enable you to add new resources to Web applications without disturbing the main deployment descriptor (web.xml)
- Use of annotations to configure resources such as servlets, instead of explicitly defining the resources in the deployment descriptor

Web Services and RESTful Services
From release 1.3 onwards NSASJ supports the JAX-WS subsystem of JBoss 7.1.2. Remote and on-platform clients can invoke services deployed in the NSASJ container by sending SOAP request messages containing XML data. JBoss 7.1.2 complies with JAX-WS specifications (JSR-224). The default Web services stack that is enabled in NSASJ is the JBoss WS Project Native Stack. With this stack enabled, NSASJ supports SOAP 1.2, WSDL 2.0 and the following WS-* specifications from W3C consortium: WS-Addressing, WS-BPEL, WS-Eventing, WS-Reliable Messaging, WS-Registries, WS-Security, WS-Policy, WS-Transactions and SAAJ.

NASJ provides JSE (POJO) and EJB EndPoints, supports standard Annotations, JAXB data bindings and SOAP Attachments.

RESTful Web Services are also available in NSASJ from release 1.3 onwards. It complies with JSR-311 specifications defined in JAX-RS. On or off-platform clients can invoke Create, Read, Update, and Delete (CRUD) operations via PUT, GET, POST, and DELETE requests in the HTTP protocol. Resources used by the server application are decoupled from their representation, so that clients can request the data in a variety of different formats. However, In RESTful services, Stateful interactions require explicit state transfer by the client and server applications.

The **NonStop Application Server for Java (NSASJ) User Guide** provides more information on how to use on Web Services and RESTful services.

Support for on- and off-platform clients
EJBs and servlets deployed in an NSASJ container can be accessed by clients running on the same or different NonStop node, as well as on an altogether different platform. Remote EJB clients use the JBoss Remoting 3 interface.

Remoting
The remoting subsystem provides the connectors to communicate with external systems. This subsystem is used in NSASJ to provide the connectivity to EJB clients that are remote (i.e., outside the NonStop system where the NSASJ server is deployed). NSASJ makes no NonStop-specific changes/configurations to this subsystem.

Continuous availability and near linear scalability
Mission-critical applications must remain Always-on. NSASJ provides this competitive advantage.

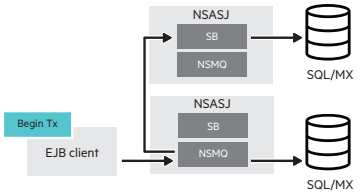
NSASJ is designed as a TS/MP Serverclass, receiving requests from clients via the Post Master within Pathsend messages. NSASJ Serverclass is configured so that both process management and dynamic scaling are handled by TS/MP. Because it is a serverclass, NSASJ is continuously available and offers near-linear scalability.

Note: One NSASJ Server definition in TS/MP can result in multiple instances that are managed by the TS/MP. This feature is unlike the JBoss domain configuration, which contains multiple servers, each with a unique server definition.

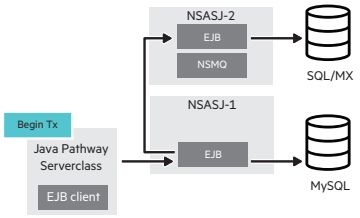
The Post Master component scales differently, based on the communication subsystem employed in a NonStop system. If the communication subsystem is TCP/IPv6, then each CPU can run only one Post Master instance. However, if IP-CIP is used as the TCP/IP provider, then multiple instances of Post Master can be started in one CPU. Post Master is configured for process management only, with TS/MP as the domain controller and Infinispan as the cache cluster.

The Infinispan cluster is configured so each cache entry has a backup copy stored in a node different from its own. A consistent hashing algorithm is used to distribute the stored data evenly across the nodes in the cluster. The number of nodes in the cluster is statically configured by the administrator.

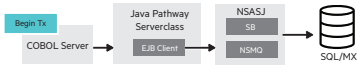
Transaction management and data integrity—integration of JTA and NS TMF
NSASJ uses the pure Java-based JTA implementation of JBoss Transaction Services (JBoss TS). In addition, NSASJ transactions are integrated with NonStop TMF. As a result, EJB and servlet application developers using NSASJ do not need to invoke any TMF or other NonStop-specific APIs. Developers can use standard JTA library calls to start/commit/abort transactions.



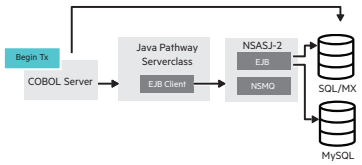
Inter-NSASJ Transaction started by on platform Java client—resource is internal



Inter-NSASJ Transaction started by on platform Java client—one resource is external



Transaction started by on platform non-Java



Transaction started by on platform non-Java process—one resource is external

All resources that can participate in a TMF transaction can become part of the global transactions started by NSASJ. NonStop servlets for JavaServer Pages, plain Java applications, and NSMQ clients can start and control XA transactions and corresponding NonStop TMF transactions using this library.

External resources such as MySQL can also participate in the JTA transaction along with NonStop Resources. In this case, when JTA transaction is committed, the JTA implementation ensures both external resource and NonStop resources are committed as a single unit of work through two phase commit. In case of failures during two phase commit, the JTA implementation recovers the external resource based on TMF transaction state to maintain the data integrity.

Global Transaction Management

NSASJ provides the ability to associate one and only one NonStop TMF transaction with an XA transaction even if multiple NSASJ instances are participating in that XA transaction. All on-platform heterogeneous applications (i.e., native applications written in C/C++, COBOL, TAL etc. and the applications deployed in NSASJ) are able to work within one global NonStop TMF transaction context. The JTA sub-system has been integrated with the NonStop TMF software to achieve this. In both cases (i.e., transaction started and controlled by a Java process or by a non-Java process) the participating data resource can be either on-platform or off-platform. The figures on the left shows some example scenarios to illustrate the feature.

Thus the design of NSASJ transaction management assures that no data is lost.

Java Persistence API

JPA is the standard API for managing persistence and object-relational mapping. JBoss uses Hibernate internally as the JPA provider.

NSASJ includes the necessary NonStop-specific tooling (such as the dialect file) to support JPA. This means that NSASJ is integration-tested with NonStop SQL/MX via its database drivers; it also means that application developers using NSASJ need only know the standard JPA APIs to persist their applications' business data. Developers do not need to be aware of the specificities of NonStop JDBC drivers (T2 and T4) and the SQL/MX database.

DataSources

NSASJ supports the DataSources subsystem provided by JBoss. Both XA and non-XADataSources can be defined, and multiple DataSource definitions can be defined if required.

A DataSource can be defined at the container level, EJB subsystem level, or application level. NSASJ allows the DataSource to be defined in the application deployment descriptor.

Security

In NSASJ, the JBoss security subsystem that supports authentication, authorization, and audit is available. Security constraints can be applied to servlets by mapping them to URL patterns and limiting the URL's access to certain roles. The NonStop Application Server for Java (NSASJ) User Guide includes references on how to configure the jboss-web.xml of the application and the security domain in domain.xml to enable this feature.

Context storage cache (In-Memory Cache)

From release version 1.4, NSASJ uses Redis cache server as its main cache mechanism to store session contexts. The data stored in the cache is also used to fulfill the NSASJ 1.4 feature Request Retry, which is described later in this document. The Redis data structure server is implemented using NonStop process-pair technology for continuous availability. Scalability of the store is achieved by sharding the data across multiple Redis cache server instances.

Up to release version 1.3, NSASJ had been using the Infinispan subsystem of the JBoss AS to store temporary-state information of the EJB and Web clients' sessions. The state of the SFSB and the servlets was saved in the Infinispan cache, and an Infinispan cluster of cache servers was configured. Each cache server was called a node. Nodes communicated with each other using JGroups over a UDP/TCP port. Infinispan used a hashing algorithm to distribute the stored data objects across the available nodes. An entry was replicated in two nodes for continuous availability. As mentioned above the Infinispan cache storage is not used from NSASJ 1.4 release onwards.

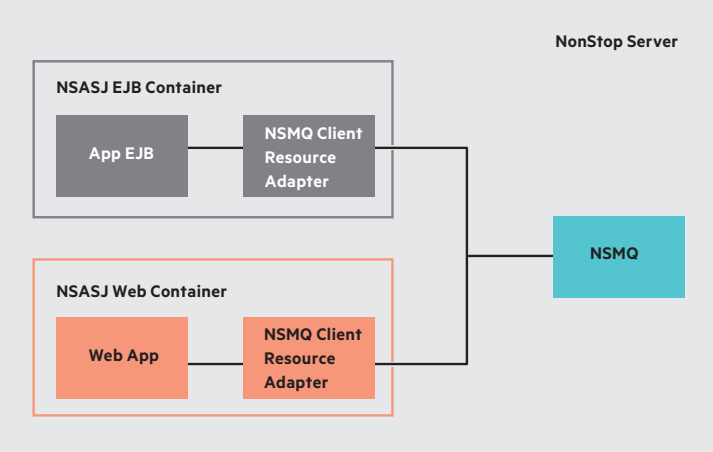


Figure 3: Integration with NSMQ

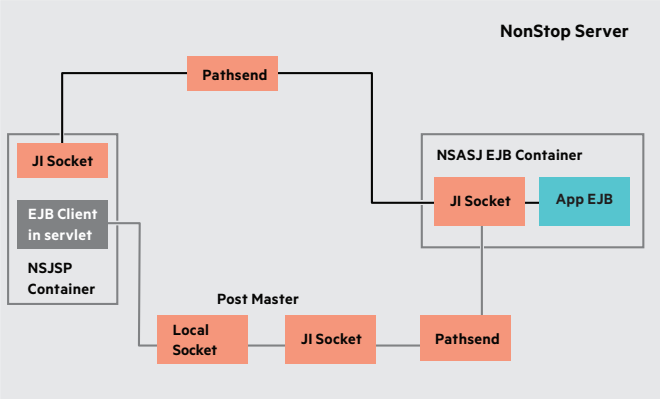


Figure 4: Integration with NSJSP

Integration of Web Container with EJB Container

In NSASJ, the Web servlets can invoke EJBs right out of the box. NSASJ packages all APIs/interfaces in a separate jar and makes the jar a dependency for the Web applications. This approach allows you to look up the modules using JNDI or CDI. Remote EJB lookups are possible by setting up outbound socket connections similar to one EJB invoking another.

Support for 64-bit mode

Sixty four bit operations are supported from NSASJ 1.2 onward. The selection of 32-bit or 64-bit mode of operation is requested at installation time. The appropriate NonStop Server for Java operation mode is based on user input.

However, application developers/users have the flexibility to start NSASJ 1.4 in a different operational mode from the one selected at installation time. Developers can simply change the value of a configuration parameter and restart the NSASJ server.

Integration with NonStop Message Queue (NSMQ)

NSASJ has been integration-tested with NSMQ 1.0, and the resource adapter provided by NSMQ is deployed within NSASJ—making NSMQ the messaging infrastructure of NSASJ.

SLSB, SFSB, and the Web application servlets use outbound connections to transfer messages to the NSMQ. NSASJ MDBs use inbound connections to receive messages from the NSMQ. The resource adapter allows the NSASJ server to manage a pool of connections and control transactions.

Propagation of XA and local transactions is supported across the outbound connections.

NonStop servlets for JavaServer Pages (NSJSP)

NSASJ has also been tested with NSJSP 7.0. NSASJ provides a client library that Web applications deployed as servlets in NSJSP can use to invoke EJBs in the NSASJ container. As shown in figure 4, NSJSP can be configured to invoke the EJB directly using the Java Infrastructure API or via the Post Master.

A JTA-compliant transaction library is also provided to the servlet application. When the application invokes an EJB from within a transaction, the context is passed onto the NSASJ server.

The security credentials are local to NSASJ and NSJSP; they are not shared between the two containers.

Request retry

The NSASJ server ecosystem implements a retry mechanism when a failure occurs while processing a client's request. Failure can be that of a CPU, an unexpected stop/interruption in NSASJ server's functions and others. If this feature is turned on the client's request will be retried and the outcome will be sent back to the appropriate client. Depending on the stage of the request processing at which the failure occurred, the actions taken will be different. More details on how to use this feature can be found in the **NSASJ User Guide**. The following points should be noted:

This feature is applicable only for the Web container (servlets) functionality of NSASJ i.e, requests coming to the applications deployed in the NSASJ Web container will only be retried. Requests to EJB applications are not retried. Please also note that HTTPS requests are not retried by NSASJ 1.4.

If this feature is turned on the transaction initiation and management will be done only by NSASJ. The client or the server application cannot start or control the NonStop transactions.

Silent Installation

From NSASJ 1.4, the administrator has an option to feed the necessary configuration parameter values into a configuration file. The set up script then uses those values during the installation process. Thus installation takes place in a non-interactive manner freeing the admin for other activities. Information for more than one NSASJ serverclass deployment can be included in the property file with Begin-End demarcations.

Using this feature multiple NSASJ installations can be done in one go and also the installation of NSASJ can be automated as part of a workflow.

nsasj_info_collector.sh

This script collects and collates useful information for the purpose of analyzing and fixing any possible defects. This will collect the necessary logging information and configuration values and archive it. This simplifies the job of customers, partners and field engineers who otherwise would need to spend time and effort to gather the information requested by development for troubleshooting issues.

Managing NSASJ

NSASJ includes CLI-based management. You can run the CLI on the NonStop platform or on a foreign platform. The CLI provides a GUI that can run on platforms such as Microsoft® Windows® or UNIX® for remotely managing the container instances. An administrator can use the CLI-based GUI to complete the following management functions:

- Query/add/update/delete container configuration elements and attributes
- Persist changes to container configurations so the changes are available across container restarts
- Query run-time statistics of infrastructure components (such as the transaction manager) and application components (such as a specific EJB)

Scripts are provided for tasks such as:

- Perform administrative tasks (such as adding a new user)
- Start, stop the container

When NSASJ is deployed on NonStop servers, there is only one server definition, but there are multiple instances of that serverclass. This is different from deployments of JBoss on other systems where there are multiple servers, each with a unique server definition in a domain. Therefore, a distinct name is assigned to each instance of the NSASJ Serverclass to make the NonStop cluster look similar to deployments on other systems. For management purposes, each NSASJ Serverclass instance appears as an individual server.

Management through the Web-based management console

From release 1.2 onward, NSASJ provides the ability to manage the EJB and Web Containers using a Web-based management console. The console is based on the JBoss Web Management Console, which has been adapted to match the NSASJ Serverclass architecture. Important functionalities offered through the GUI include:

- Deploy, disable/enable, and remove applications
- Display runtime server status for EJB and Web Containers
- Display and modify configurations for the NSASJ server, JVM, and network interfaces
- Display and modify the profiles for the subsystems supported by NSASJ

JMX Management Interface

A new JMX-PM (Java Management Extensions-Post Master) module is available from NSASJ 1.4. JMX-PM is a TS/MP serverclass. It listens on a unique port for each NSASJ server instance for management control messages. This enables multiple NSASJ server instances to be simultaneously monitored independent of each other.

With JMX, you can monitor JVM details like number of classes loaded/unloaded, memory utilization, number of threads, JVM summary etc. However, you cannot enable/disable applications and view/modify servers' configurations. JMX is for monitoring purpose while the Web based management console is for administrative operations.

You can find complete details in the NonStop Application Server for Java (NSASJ) User Guide available in **hpe.com/info/nonstop-docs**.

Application deployment

In NSASJ, you use the CLI to deploy applications as EAR, JAR, WAR, or RAR packages. You can deploy, undeploy, redeploy, enable, disable, and remove applications. In addition, you can integrate application development environments with NSASJ using Maven, just as you would with JBoss AS.

System requirements of the host

- HPE Integrity NonStop X servers or HPE Integrity NonStop BladeSystem servers
- NonStop operating system release version L15.02/J06.16 or later Open System Services (OSS) environment (if using NSJ 7 and TS/MP 2.5)
- NonStop operating system release version L16.05/J06.20 or later Open System Services (OSS) environment (if using NSJ 8 and/or TS/MP 2.6)

Note: Please refer to the latest edition of Software Products Maintenance List (SPML) to know the support status for different RVU versions.

- NonStop TCP/IPv6 or IP-CIP for networking
- NonStop Server for Java 7 (NSJ 7) or NonStop Server for Java 8 (NSJ 8)

Note: For 64-bit mode of operation one of the following combinations must be installed as appropriate:

- Both T2766 H70 ^ACN and T2866 H70 ^ACN or later versions
- Both T2766 L70 and T2866 L70 or later versions
- Both T2766 H80 and T2866 H80 or later versions
- Both T2766 L80 and T2866 L80 or later versions
- NSJ Infrastructure (JI)—T2966H70 ^AAF and T2966L70 ^AAD or a later version (if using NSJ 7)
- NSJ Infrastructure (JI)—T2966 H80 or T2966 L80 or a later version (if using NSJ 8)
- NonStop TS/MP version 2.5 or later
- TMF version 3.7 or later
- NonStop Message Queue version 1.0 (NSMQ 1.0) or later (optional)
- NonStop servlets for JavaServer Pages 7.0 (NSJSP 7.0) or later (optional)

Note: Orderable name for NSJSP 7.0 is NSJSP 6 Update 4

- NonStop SQL/MX version 3.2.1 or later (optional)

Product ordering information

- For J-series—Order QSJ87V1 NonStop Application Server for Java
- For L-series—Order BE309AC HPE NonStop App Server for Java SW
- Purchase the QSJ87V1 or BE309AC license on a per-CPU basis

Related HPE offerings

Customer technical training
Gain the skills you need with training from Hewlett Packard Enterprise. Accelerate your technology transition, improve operational performance, and get the best return on your HPE investment. Training is available when and where you need it, through flexible delivery options and a global training capability. **hpe.com/ww/learnnonstop**

HPE Technology Services

HPE Technology Services help build an infrastructure that is reliable, highly available, and rooted in best practices. Hewlett Packard Enterprise recommends the following services:

HPE Critical Service: High-performance reactive and proactive support designed to minimize downtime. It offers an assigned support team, which includes an Account Support Manager (ASM). This service offers access to HPE's Global NonStop Solution Center, 24x7 hardware and software support, six-hour Call-to-Repair commitment, enhanced parts inventory, and accelerated escalation management.

HPE Proactive 24 Service: Provides proactive and reactive support delivered under the direction of an ASM. It offers 24x7 hardware support with four-hour onsite response, 24x7 software support with two-hour response and flexible call submittal.

HPE Support Plus 24: Provides reactive hardware and software support with remote problem diagnosis, four-hour onsite response, replacement parts. The software support includes installation advisory support, software updates for HPE and selected third-party software products.

HPE Installation and Startup Service: This service provides efficient and effective deployment of HPE NonStop system.

For more information, visit **hpe.com/services**

Learn more at
hpe.com/info/nonstop

Java™ COMPATIBLE

Sign up for updates

Hewlett Packard Enterprise

4AA4-6102ENW, September 2016, Rev. 6