V3 EJB Test One Pager

Overview

1. Introduction
2. EJB Testing Scenarios
    2.1 EJB Lite Features
    2.2 API only in Full EJB3.1
3. Document Review
4. Reference documents

---

1. Introduction

This document describes test specification for **EJB 3.1** supported by GlassFish v3 FCS.

1.1. Names and e-mail of Document Authors
Ming Zhang: Ming.Zhang@sun.com

1.2 Revise History:
01/10/2009 1.0 revision.
03/25/2009 1.1 revision.

1.3 Test Priorities:

- **(P1)** A simplified Local view that provides Session Bean access without a separate Local Business interface.
- **(P1)** Packaging and deployment of EJB components directly in a .war without an ejb-jar.
- **(P1)** A Singleton session bean component that provides easy access to shared state, as well as application
  startup/shutdown callbacks.
- **(P1)** Asynchronous session bean invocations.
- **(P1)** Automatically created EJB Timers.
- **(P1)** The definition of a lightweight subset of Enterprise JavaBeans functionality that can be provided within Java EE Profiles such as the Java EE Web Profile.
- **(P2)** A portable global JNDI name syntax for looking up EJB components.
- **(P2)** An embeddable API for executing EJB components within a Java SE environment.
- **(P3)** Stress testing of timers.
- **(P3)** Calendar based EJB Timer expressions.

2. EJB Testing Scenarios

EJB in V3 will be tested in both Web Distribution and full Glassfish Distribution. V3 Web Distribution contains EJB Lite.
Glassfish Distribution contains EJB lite plus APIs only available to the full Glassfish Distribution. For the V3 Web Distribution,
the tests will be limited to using the APIs that are part of EJB 3.1 Lite. For the full Glassfish Distribution, the tests will be for
the EJB 3.1 Lite + the APIs that are only in the full Glassfish Distribution.

2.1 EJB Lite Features

EJB 3.1 Lite will be available in V3 web distribution. GF V3 supports war deployment for EJB jar.

2.1.1 Local view of EJB injection with war packaging of EJB components
Use stateless session bean with local business interface
Inject @EJB for Stateless session bean in a servlet,
Inject @Resource for DataSource, UserTransaction in a stateless session bean,
Retrieve DataSource, UserTransaction in the servlet via the ejb.
Synchronous method invocations only

2.1.2 Stateless and Stateful session Bean with no-interface view.
Use stateless session bean without local business interface
Has similar annotation tests as 2.1.1

2.1.3 EJB and JPA with CMT
The stateless session bean is used without a local business interface.
Injection of EntityManager is in stateless session bean, which is thread safe.
The operations of persisting, querying, removing entity are performed
in the stateless session bean with container managed transaction.

2.1.4 EJB and JPA with BMT
The stateless session bean is used without a local business interface.
Injection of EntityManager is in stateless session bean, which is thread safe.
The operations of persisting, querying, removing entity are performed
in the stateless session bean with bean managed transaction.

2.1.5 Interceptors in Stateless Session Bean with Local Interface
Default interceptor in ejb-jar.xml for all beans including a *SLSB*
External interceptor with @Interceptor in a *SLSB*
Internal interceptor with @AroundInvoke in a *SLSB*
<exclude-default-interceptors> in ejb-jar.xml for a *Singleton*.
That *Singleton* provides an access to shared state that tracks interceptor invocation.
When *SLSB* is accessed, all 3 types of interceptors are invoked.

2.1.6 Interceptors in Stateless Session Bean without Interface
Default interceptor in ejb-jar.xml for all beans include the SLSB
External interceptor with @Interceptor in the SLSB
Internal interceptor with @AroundInvoke in the SLSB
@ExcludeDefultInterceptors for a Singleton.
That *Singleton* provides an access to shared state that tracks interceptor invocation.
When the SLSB is accessed, all 3 types of interceptors are invoked.

2.1.7 Singleton Startup
EJB injection for singleton *@EJB private BeanRoot root*
JNDI lookup for singleton *"java:global[/<app-name>]/<module-name>/<bean-name>"*
Singleton Initialization with *@Startup*

- For a singleton with *@Startup*, *PostConstruct* is invoked during deployment
- For another singleton without *@Startup*, *PostConstruct* is invoked when it is accessed.

2.1.8 Singleton DependsOn
DependsOn with Startup

- *@Singleton BeanRoot* has *@Startup* and *@DependsOn({"BeanLeaf"})*
- *PostConstruct* is invoked during deployment in the order of *BeanLeaf*, then *BeanRoot*.

DependsOn without Startup

- *BeanA* and *BeanB* depend on *BeanC*, and *BeanC* depends on *BeanD*
- When application access *BeanA, BeanB, BeanC,* then *BeanD,* the *PostConstruct* is invoked in the order of *BeanD, BeanC, BeanA*, then *Bean B*.

2.1.9 Singleton and JPA with Container Managed Transaction

JNDI lookup of an *EntityManager* is in a singleton,
which is not thread safe, but is OK with 1 client access during the test.
The operations of persisting, querying, removing entity are performed
in the singleton with container managed transaction.

2.1.10 Singleton with Bean Managed Concurrency

To ensure a synchronized business method of a singleton is processed sequentially when multiple clients access it.

- A BMCSingleton has a synchronized business method.
- A multi-threads java client accesses a BMCServlet.
- BMCServlet invokes a synchronized business method in BMCSingleton.
- BMCSingleton also invokes the business method in DateSingleton.
- TransactionAttributeType.NOT_SUPPORTED is specified for DateSingleton and the a synchronized business method in BMCSingleton.

2.1.11 Singleton with Container Managed Concurrency

To ensure a synchronized business method of a singleton is processed sequentially when multiple clients access it.

- A CMCSingleton has a business method.
- A multi-threads java client accesses a CMCServlet.
- CMCServlet invokes a synchronized business method in CMCSingleton.
- CMCSingleton also invokes the business method in DateSingleton.

TransactionAttributeType.NOT_SUPPORTED, REQUIRED and REQUIRES.NEW are specified for DateSingleton and the a synchronized business method in CMCSingleton.

2.1.11 Interceptors in Stateless Session Bean with Local Interface

Default interceptor in ejb-jar.xml for all beans including a *SLSB*
External interceptor with @Interceptor in a *SLSB*
Internal interceptor with @AroundInvoke in a *SLSB*
<exclude-default-interceptors> in ejb-jar.xml for a *Singleton*.
That *Singleton* provides an access to shared state that tracks interceptor invocation.
When *SLSB* is accessed, all 3 types of interceptors are invoked.

2.1.12 Interceptors in Stateless Session Bean without Interface

Default interceptor in ejb-jar.xml for all beans include the SLSB
External interceptor with @Interceptor in the SLSB
Internal interceptor with @AroundInvoke in the SLSB
@ExcludeDefultInterceptors for a Singleton.
That *Singleton* provides an access to shared state that tracks interceptor invocation.
When the SLSB is accessed, all 3 types of interceptors are invoked.

2.1.13 Redeployment testing of various EJB 3.1 applications.

Redeployment is a product-specific feature not tested by CTS. It will be tested with some applications using of EJB 3.1 features.

2.1.14. SFSB passivation/activation combined with new EJB 3.1 features.
By definition, passivation/activation are product-specific behaviors. We should ensure that the new EJB 3.1 features applied to SFSBs are passivated and activated correctly :

- SFSB with a no-interface view
- SFSB holding an ejb reference to a Singleton

2.2 API Available only in Full EJB 3.1
Some EJB features (APIs) are available only in V3 Glassfish Distribution and not included in EJB 3.1 lite.

2.2.1 Asynchronous session bean invocations.
To ensure a Asynchronized business method of a stateless session bean is processed in parallel when multiple orders access it.

- Use @Asynchronous to asynchronize business methods in a stateless session bean.
- The returns will be stored in generic Future array (String, Double).
- Use HelloASyncServlet to invoke business methods the stateless session bean and get the returns from Future array.
- A java client to accesses a HelloASyncServlet and return test results.

2.2.2 Multi-thread Asynchronous session bean invocations.
To ensure a Asynchronized business method of a stateless session bean is processed in parallel when multiple clients and orders access it.

- Use the multi-thread java client to accesses a HelloASyncServlet and return test results.

2.2.3. Test EJB ear packaging scenarios.

2.2.4. Develop EJB timer test. Configuring an external timer database instead of the default one and run EJB timer tests.

2.2.5. Stress testing of timers
CTS tends to focus on applications with either one or a very small timer of timers. It would be good to add applications with a large number of timers and/or very fine-grained timeouts.

2.2.6. Testing of the developer experience in the face of common errors.
If an EJB application contains an invalid EJB (e.g. one with a @Startup Singleton whose @PostConstruct throws an exception), what's the behavior? Is the error correctly reported during deployment?

2.2.7. Stand-alone (plain Java) client access to the new EJB 3.1 features (Remote Async, Singletons, etc).
Stand-alone clients are not defined by the spec.

2.2.8. Regression tests to ensure that our V2 and earlier product-specific global JNDI names continue to work.
The V2 glassfish-specific global JNDI naming approach (defined here : https://glassfish.dev.java.net/javaee5/ejb/EJB_FAQ.html#SessionBeanGlobalJNDINameAssignment)

3. Features Tested in Other Modules:

3.1 Refer to SQE security test spec for EJB Security testing.
3.2 Refer to JMS test plan for MDB testing.
3.3 Refer to Embedded GF test spec for EJB+Embedded GF testing.

4. Reference Documents

4.1 GF V3 EJB Container Half Pager
4.2 EJB Core Contracts and Requirements, JSR 318 EJB3.1