1. Introduction

1.1. Project/Component Working Name:

SJSAS 9.1, Support for JDBC 4.0 in JDBC RA, RFEs

1.2. Name(s) and e-mail address of Document Author(s)/Supplier:

Jagadish Ramu : Jagadish.Ramu@Sun.COM

1.3. Date of This Document:

08/09/06 - Created

10/10/06 - Updated with changes related to JDBC 4.0 Specification & 1 RFE

2. Project Summary

2.1. Project Description:

JDBC Resource Adapter to support JDBC 4.0 features when Application Server is used in JDK 1.6 environment.

2.2. Risks and Assumptions:

JDBC 4.0 is in proposed final draft. Changes to the draft may affect the JDBC RA's support.

3. Problem Summary

3.1. Problem Area:

To support JDBC 4.0 for Application Server environment.

API for application to mark a connection as bad

3.2. Justification:

1) SJSAS will be first among the application servers to support JDBC 4.0 features

2) It is necessary to support JDBC 4.0 at the same time when users start using Application Server running on JDK 1.6

3) Sometimes application may not prefer to use "validation" mechanism provided by connection-pool and is aware of connections becoming bad. In such cases, marking a connection as bad will help the pool to remove the bad connection.

4. Technical Description:

4.1. Details:

JDK 1.6 incorporates JDBC 4.0 features. Application Server when run using JDK 1.6 need to support JDBC 4.0 features.

JDBC RA must also provide backward compatibility when run using JDK 1.5.

4.2. Bug/RFE Number(s):

4.3. In Scope:

4.4. Out of Scope:

4.5. Interfaces:

4.5.1 Exported Interfaces :

a) All wrappers that will be exposed to JDBC API users in Application Server Environment

Interface: JDBC 4.0
Stability: Standard
Exporting Project: JDBC Resource Adapter of Application Server

b) com.sun.appserv.jdbc.DataSource Interface exposed to applications will have the following method :
public void markConnectionAsBad(Connection)

Applications using this feature :

```
com.sun.appserv.jdbc.DataSource ds=
        (com.sun.appserv.jdbc.DataSource)context.lookup("dataSource");
Connection con = ds.getConnection();
Statement stmt = null;
try{
         stmt = con.createStatement();
         stmt.executeUpdate("Update");
}catch(BadConnectionException e){
        dataSource.markConnectionAsBad(con) //marking it as bad for removal
}finally{
        stmt.close();
        con.close(); //Connection will be destroyed during close.
}
```

4.5.2 Imported interfaces

Standard : based on JDBC 4.0 API

Interface: JDBC 4.0
Stability: Standard
Exporting Project: JDBC 4.0

4.5.3 Other interfaces (Optional) : None

4.6. Doc Impact:

Developer Guide
Administration Guide

4.7. Admin/Config Impact: None

4.8. HA Impact: None

4.9. I18N/L10N Impact: None

4.10. Packaging & Delivery: None

4.11. Security Impact: None

4.12. Compatibility Impact

Users must be able to use JDBC 3.0 API when Application Server is using JDK 1.5.

4.13. Dependencies: JDBC 4.0 API

5. Reference Documents:

JDBC 4.0 Specification (JSR 221)
JDBC 4.0 API
RFE 6473757 - RFE: Provide an API to allow an Application to flag that a connection is bad when it is returned

6. Schedule:

6.1. Projected Availability:

Aligned with Overall SJSAS 9.1/GlassFish V2 Schedules

7. Sample Code exhibiting JDBC 4.0 features in Application Server Environment

NOTE: Ease of Development (EOD) features (QueryObjectGenerator) are removed from JDBC 4.0. Hence the following sample is not applicable.

----------Query ----------

```
import java.sql.*;
import javax.sql.*;

    interface MammalQueries extends BaseQuery {
        @Select(sql = "select lastname, firstname, age, weight, description from mammals",
                allColumnsMapped = true)
        DataSet<Mammal> getAllMammals();

        @Select(sql = "select * from mammals where firstName= ?1 and lastName = ?2")
        DataSet<Mammal> getMammalsByName(String fName, String lName);

        @Update(sql = "update mammals set age=?1 where lastName = ?2")
        int setMammalAge(int age, String lName);
    }
```

```
----------------------------------------------------Java Bean----------------------------------------------------
public class Mammal implements java.io.Serializable{
        public String firstName;
        public String lastName;
        public int age;
        public int weight;
        public String description;

        public String toString(){
                StringBuffer buffer = new StringBuffer();
                buffer.append("[ first name : " + firstName + " ] ");
                buffer.append("[ last name : " + lastName + " ] ");
                buffer.append("[ age : " + age + " ] ");
                buffer.append("[ description : " + description  + " ] ");
                buffer.append("[ weight : " + weight + " ] ");

                return buffer.toString();
        }
}


----------------------------Stateful Session Bean using Query Object-----------------------
import java.sql.*;
import javax.sql.*;
import java.util.*;
import javax.ejb.*;
import javax.annotation.*;


@Stateful
public class SfulEJB implements Sful
{
        @Resource private SessionContext ctxt_;
        @Resource (name="java:comp/env/DataSource", mappedName="jdbc/s1qeDB")
        private   DataSource ds;
        MammalQueries myQueries ;
        boolean initialized = false;

        public void init(){
            if(!initialized){
                try {
                        myQueries = ds.createQueryObject(MammalQueries.class,ds);
                        initialized = true;
```

```
        }catch (Exception e){
                e.printStackTrace();
        }
    }
}

public ArrayList<Mammal> getAllMammals() throws Exception {

    init();

    ArrayList<Mammal> mammals = new ArrayList<Mammal>();
    DataSet<Mammal> rows = myQueries.getAllMammals();

    System.out.println("--------------getAllMammals----------------------");

    for (Mammal m : rows) {
        System.out.println(m.toString());
        mammals.add(m);

    }

    System.out.println("-------------------------------------------------");
    rows.close();
    return mammals;
}

public ArrayList<Mammal> getMammalsByName(String firstName, String lastName) throws Exception{

    init();
    DataSet<Mammal> rows = myQueries.getMammalsByName(firstName, lastName);
    ArrayList<Mammal> mammals = new ArrayList<Mammal>();

    System.out.println("--------------getMammalsByName--------------------");

    for (Mammal m : rows) {
        System.out.println(m.toString());
        mammals.add(m);

    }

    System.out.println("-------------------------------------------------");
    rows.close();
    return mammals;
}
```

```
    public void setMammalAge(int age, String lastName) throws Exception {

        init();
        int rowsAffected = myQueries.setMammalAge(age, lastName);

        System.out.println("----------setMammalAge----------");
        System.out.println("Rows Affected  : " + rowsAffected);
        System.out.println("------------------------------------------------");

    }
}
```