[TOPCODER]
SOFTWARE

ID Generator Requirements Specification

1. Scope

The ID Generator component provides high performance key generation services.

1.1 Overview

The ID Generator component provides high performance key generation services. It generates unique long IDs while minimizing both network and database overhead. The ID Generator works by implementing a High/Low scheme to generate unique IDs. It also allows IDs to be generated uniquely for a user-defined value.

This new version fixes one bug / issue and also adds some additional functionality.

1.2 Logic Requirements

1.2.1 ID Generator Plug-in Fix

- This component was supposed to have a pluggable backend that actually provided the id generation service. It currently does not support that correctly. This will need to be fixed so that a new plug-in (defined in section 1.2.2) can be added.

1.2.2 Oracle Sequence Plug-in

- ID Generator currently has its own default method for generating ids. A new plug-in will be designed that will allow the id's to be generated from an Oracle sequence.
- Some sort of mappings will need to be set that will map the ID Generator's id name to the actual Oracle sequence that will be used to generate the identifier.

1.2.3 Support for Large Identifiers

- The current version of ID Generator only supports `long` identifiers. Support for `java.math.BigInteger` identifiers needs to be added. The method signature to support this should be:

```
public java.math.BigInteger getNextBigID() throws IDGenerationException,
IDsExhaustedException;
```

- This method will only be used by plug-ins especially designed for `BigInteger` data types. All current plug-ins (the default and new Oracle sequence plug-in) do not need to worry about storing their ID's in `BigInteger` form. They should simply wrap the long id in a `BigInteger` if a client application calls this new method.

1.2.4 Database Access

- Database access should now be done through the DB Connection Factory component.

1.3 Required Algorithms

No new algorithms are defined.

1.4 Example of the Software Usage

TopCoder might use the ID Generator in the generation of new identifiers for an artificial primary key to a database table. Each time a new primary key is required – a call to the ID Generator returns a new primary key. If the application prefers to allow separate sets of ID's generated per table, the application can call the ID Generator passing the table name as the user defined value. A separate ID is then maintained by table name.

1.5 Future Component Direction

None specified.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

Must meet original ID Generator requirements.

2.1.2 External Interfaces

Must meet original ID Generator requirements.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.3, Java1.4

2.1.4 Package Structure

com.topcoder.util.idgenerator

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Must meet original ID Generator configuration requirements, keeping in mind that the configuration options now need to be able to be set programmatically.
- The new Oracle Sequence Plug-in needs to have the sequence name to id name configurable.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

Must meet original ID Generator requirements.

3.2.2 TopCoder Software Component Dependencies:

- DB Connection Factory 1.0
- Configuration Manager 2.1.3

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

Must meet original ID Generator requirements.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.