One Pager: New Connection Pool Features and Attributes

Table of Contents

1. Introduction

1.1. Project/Component Working Name

Sun Java System Application Server 9.1, New connection pool features

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

Name	Email
Kshitiz Saxena	kshitiz.saxena@sun.com
Jagadish Ramu	jagadish.ramu@sun.com

1.3. Date of This Document

Date	Version	Author	Remarks

Aug 7, 2006	1.0	Jagadish Ramu, Kshitiz Saxena	Initial Draft
Aug 9, 2006	1.1	Jagadish Ramu, Kshitiz Saxena	Incorporated Binod's suggestions
Aug 14, 2006	1.2	Jagadish Ramu Kshitiz Saxena	Incorporated Siva's suggestions

Oct 3, 2006	1.3	Jagadish Ramu Kshitiz Saxena	• Features Dropped for 9.1 :
			  ○ Clustered Pool
			  ○ Merging Connection Pool and Resource as a single command
			• New properties :
			  ○ ConnectionLeakReclaim
			  ○ MaxConnectionUsageCount

Oct 27, 2006	1.4	Kshitiz Saxena	• Changed properties to attributes • Added attribute wrap-jdbc-objects • Added section containing old properties now changed to attributes • Removed section JDBC connection pool additional properties • Removed old dtd under external interface section. Provided link to DTD changes for connection pool • Updated CLI/GUI support for
Nov 17, 2006	1.5	Jagadish Ramu	• Incorporated suggested clarifications from Chris Casso • Clarified statement-timeout behavior, order of precedence • Corrected match-connections default value for connector-connection-pool & jdbc-connection-pool • Corrected older property to new attribute order of precedence
Dec 10, 2006	1.6	Jagadish Ramu	• Corrected default value of "statement-timeout-in-seconds" of jdbc-connection-pool.

2. Project Summary

2.1. Project Description

Addition of connection pool features to provide

- ~~Load-balancing and fail-over capabilities for connections~~
- Ease of use
- Diagnosability of connection leaks

2.2. Risks and Assumptions

~~The clustered-pool feature will be implemented and tested to work only with Oracle RAC. It can be extended later to work with other systems.~~

3. Problem Summary

3.1. Problem Area

- Absence of load-balancing and fail-over capabilities for connections to clustered enterprise systems.
- Inability to trace connection leaks
- No connection retrial at the time of connection creation in case of failure
- No efficient mechanism to validate connection to enhance performance
- Ease of connection pool creation and configuration
- No mechanism to switch off connection pooling in ACC
- To provide feature parity with other Application Servers

3.2. Justification

The introduction of proposed features will highly enhance the user experience. The impact of these changes will be at many places. The support of clustered-pool which will provide enterprise features like load-balancing and failover feature for connection to the user. The clustered-pool feature would lead to enhanced scalability and availability of connections to clustered external system. From the ease of creation of connection pool, to ability of connection leak tracing, the user will have better user experience.

4. Technical Description

4.1. Details

~~ClusteredPools~~

~~ClusteredPools is a set of connection pools, each pool is configured against a node of a cluster in an external system. For example, Oracle Real Application Cluster [RAC] - ClusteredPools can be used with Oracle RAC. In a typical setup, each pool in the clustered-pool will be configured to talk to one particular node of the cluster in an external system. To enable pool clustering, user should provide a list of connection pools, instead of selecting single connection pool, at the time of resource creation. The following features are provided by clustered-pool :~~

- ~~**Load-Balancing** : Connection requests from the application can be routed to different nodes of the cluster. This sub-feature helps to share the load across different nodes in the external system. The load-balancing policy chosen for this release is round-robin. New policies will be introduced in future releases.~~
- ~~**Fail-Over** : When a connection request to one of the node fails, ClusteredPool will stop using the pool associated with that node, and switch to other pools. This sub-feature enables application server to aid in using the high-availability features provided by external system.~~

~~The connections are load-balanced and failed-over to healthy nodes only during connection request from the application. There is no fail-over support for connections of in-flight transactions. ClusteredPool uses the connection-validation feature of the connection pool. When a connection is requested by the application, clustered-pool will choose a pool from its list and validates the connection. If the validation is successful, connection is returned to the user. If the connection is invalid, clustered-pool will fail-over to other available pools. All connections in the failed-pool will be dropped. Depending on the configuration, clustered-pool will keep polling the failed-node at regular intervals. This interval can be specified using property *PollInterval.* The default value will be 60 seconds. The appserver will keep polling the failed node, till it comes up. Then pool will be re-created and new connection requests will be routed to this pool as well. There is no timeout associated with node-polling. Hence it may happen that appserver may keep polling a failed node throughout its lifetime, if that node never comes up.~~

~~The clustered-pool feature will be available for JDBC and connector resources. Refer to Admin/Config Impact section for CLI/GUI changes~~

New Connection Pool Config Attributes

The proposal is to introduce several attributes to configure connection pool at the time of creation. The list of the attributes being proposed and their detailed description are as follows :

- **Connection Leak Tracing** - As the name suggests this attribute will be used to switch on or off the leaking tracing for a connection pool. By default, there will be no leak tracing. However if enabled, the connection pool will provide data to the user wrt application leaking the connections. An application is said to be leaking connection if it acquires a connection, and does not close it within the specified time period.

  All leak tracing logs will be dumped to server logs. The user can search through the logs via GUI and find out connection leak traces for a pool. The utility to search through logs already exists in GUI.

  A new monitoring statistics to reflect number of connection leaks corresponding to a pool is being added. The user can find this new statistics along with other monitoring statistics, once monitoring is switched on for Connector/JMS or JDBC.

  The connection leak tracing will be enabled using attribute ***connection-leak-timeout-in-seconds.*** By default this attribute is set to zero, i.e., connection leak tracing is switched off by default. When this attribute is set to positive non-zero value, connection leak tracing is enabled.

  Another attribute ***connection-leak-reclaim*** is being introduced to reclaim leaking connection. By default it is set to false, which implies no connection reclaim. If enabled, connection will be re-usable (put back to pool) after connection-leak-timeout-in-seconds occurs.

- **Connection Creation Retry** - As of now, when application requests for connection
  - Either a connection provided from pool if one free connection exists
  - Or one free connection is created, if pool size is less that max pool size, and returned to application

  In second case, if connection creation fails in first attempt, due to any reason, an exception is thrown to application and application should handle connection creation failure and should re-request for connection.

  With the introduction of attribute ***connection-creation-retry-attempts,*** the connection retry can be enabled at the connection pool level. By default, this attribute is set to zero and connection retry is turned off. When set to positive non-zero value, the connection creation will not fail after first attempt. The connection creation will be retried for finite number of times, as provided via this attribute, before application if notified via exception about connection creation failure. Also user can specify the interval of connection creation retrial using attribute ***connection-creation-retry-interval-in-seconds***. By default, this attribute is set to 10 seconds.

- **Validate Atmost Once Period** - Lets refer to first case mentioned in previous attribute, i.e., a free connection being returned from pool to application. There is a attribute ***isconnectvalidatereq,*** to enforce connection validation, before connection is returned to the application. The validation is a costly operation to perform for each and every connection requested by applications from pool. However, switching off connection validation completely is also not preferable as application may get stale, invalid connections. So, the solution is to find a midway, i.e., to reduce the frequency of connection validation. A recently validated connection may be safely given to requesting application without validation. Though a limit needs to be defined on what will be considered as acceptable time period in which connection validation is not required. This limit can be defined using attribute ***validate-atmost-once-period-in-seconds.*** A connection will be validated only if it has not been validated before in the last specified validate-atmost-once-period-in-seconds seconds. Otherwise it is returned without validation to requesting application. By default, this attribute will be set to 0, so that connection validation works as it works now.

- **Statement Timeout** - A long running jdbc query executed by an application may leave it in hanging state, unless a timeout is explicitly set on the statement. As of now, there is no generic way in which statement timeout can be set on all statements created using a connection. After introduction of the attribute ***statement-timeout-in-seconds***, all statements created using the pool, with this attribute set, will share the specified statement timeout. During statements creation, query timeout will be set according to the value

specified in this property. If user explicitly sets a value for the statement, it will take precedence. All queries will automatically timeout if not completed within specified period. This will also bring feature parity with competitive product. By default, this value will be -1, disabled.

- **Max Connection Usage Count -** The attribute ***max-connection-usage-count*** can be specified, so that connections will be re-used by the pool for the specified number of times after which it will be closed. This will be useful to avoid statement-leaks. Default value is 0, which implies the feature is not enabled.

- **Wrap JDBC Objects** - The attribute ***wrap-jdbc-objects*** can be set to true, so that application will get wrapped jdbc objects for Statement, PreparedStatment, CallableStatement, ResultSet and DatabaseMetaData. Default value is false, which implies application will not get wrapped objects. This is done so, as wrapped objects has performance impact.

Properties changed to attributes

The following set of properties are changed to attributes in this release.

- ***LazyConnectionEnlistment*** property changed to ***lazy-connection-enlistment*** attribute. Default value is false.
- ***LazyConectionAssociation*** property changed to ***lazy-connection-association*** attribute. Default value is false.
- ***AssociateWithThread*** property changed to ***associate-with-thread*** attribute. Default value is false.
- ***MatchConnections*** property changed to ***match-connection*** attribute. ~~Default value is true~~. Default value is true for connector-connection-pool, false for jdbc-connection-pool.

For backward compatibility both properties and attributes will be supported. ~~The value set via attribute will take precedence over value set via property.~~ The value set via property will take precedence over value set via attribute, which will ensure that existing customers using "property" will not be affected. This support for properties will be removed in future releases.

Switching off connection pooling in ACC

The ACC share the same flow as appserver, for a connection creation. Thus connection pooling starts working in ACC, when a connection is created, even though it is not desired. So an option need to be provided in ACC to switch off connection pooling if desired. A system property, namely ***com.sun.enterprise.Connectors.SwitchoffACCConnectionPooling,*** to that effect will be introduced. If set to true, the connection pooling will be turned off in ACC. By default the value of the property will be false.

~~JDBC Connection Pool Additional Properties~~

~~The connection pool configuration requires the user to define several properties when creating a connection pool. The list of properties is derived from a set of mandatory properties, sun-specific properties and then driver specific properties. This list becomes huge as many jdbc driver have long list of configurable properties. A user new to jdbc will find it difficult to configure jdbc connection pool with huge list of properties. There is a need to differentiate between a new user and an experienced user. The experienced users can use available properties to their advantage while a new user can work with minimal set of mandatory properties.~~

~~As per JDBC specifications, there are nine mandatory properties needed to connect to any database. Those are as follows:~~

- ~~user~~
- ~~password~~
- ~~port~~

- ~~serverName~~
- ~~databaseName~~
- ~~datasourceName~~
- ~~networkProtocol~~
- ~~roleName~~
- ~~url~~

~~To ease the creation of connection pool, user can be provided with only above mentioned list of properties to connect to database. The other set to properties, i.e., sun-specific properties and driver specific properties list can be provided on demand. The different set of properties can be provided either by traversing through multiple wizard-style pages or via collapsible table in GUI. Breaking the complete set of properties into three different sets makes is more understandable to user, as well as provides a better user experience.~~

~~A similar user experience must be provided to user creating connection pool using CLI. For example, sun-specific properties can be specified at command line using *sun-specific-property* attribute instead of *property* attribute. The changes to GUI/CLI are specified in Admin/Config Impact section.~~

~~There are three properties namely, *LazyConnectionEnlistment*, *LazyConnectionAssociation* and *AssociateWithThread* which are being taken as value for attribute *property* from 9.0/9.0UR1. To be backward compatible, there will be support to specify these properties as value for attribute *property* as well as *sun-specific-property* in 9.1. In future releases, this support will be removed.~~

~~Merging Connection Pool and Resource~~

~~A user writes a simple application to access database, to perform some basic operation on database. The application is deployed and as a setup step, user has to create jdbc resource as defined in the application. The user need to create a connection pool and then create required resource using the connection pool. Even after the user is done with application execution, the pool will exist and maintain itself to steady pool size, eating up the system resources. If user is provided with an option to create jdbc resource, without creation of connection pool, this overhead of connection pool creation and maintenance can be reduced. However, this is not the solitary case where connection pooling is not needed. There are some resource adapters, for example JAXR resource adapter, which implement their own pooling and caching mechanism, for better performance. In such cases, it will be beneficial to use resource adapter's connection pooling, as it may perform better than appserver's connection pooling. However being hidden behind appserver connection pooling, user may never be able to gain any performance benefit out of it.~~

~~Not just with performance perspective, the user will have better user experience, if there is a provision to create a jdbc resource without a need to create a connection pool. This can be achieved by linking resource creation with pool creation. Since user does not need a connection pool, only required properties are entered by user to connect to the external system. Also at the time of resource creation, the user may provide required connection pool parameters, and both resource and connection pool can be created using single command. The samples of CLI commands are provided in Admin/Config Impact section.~~

4.2. Bug/RFE Number(s)

1. CR 6436557 - Change request for Sun AppServer to provide load balancing for jdbc connections
2. CR 6436556 -Change request for Sun AppServer to support Oracle 10g Rac using Oracle 10g Type 4 jdbc thin driver
3. CR 6472065 - JDBC ReclaimTimeout - Enhancement Request for JDBC Connection Pool management
4. CR 6473760 - RFE: Provide an option to physically close a connection after it has been acquired X times.

4.3. In Scope

1. ~~A set of connection pools can be used transparently for single jdbc, connector-resource.~~

2. The connection leak tracing will be based on mechanism listed in document. This setting will be dynamically configurable. The appserver will pick the value on the fly and provide data to the user.
3. ~~The earlier model of connection pool creation and resource creation will co-exist. This will ensure backward compatibility.~~

4.4. Out of Scope

1. ~~Ongoing transactions will not have transparent failover.~~
2. The connection leak tracing will not be exhaustive. It will be based on mechanism defined in the documents. It can be incrementally enhanced in next releases.
3. ~~The concept of connection pool is not being removed completely. It may be deprecated in future releases.~~

4.5. Interfaces

4.5.1 Exported Interfaces

- ~~DTD changes associated with clustered pool~~

  ~~554a555,560~~
  ~~> attributes :~~
  ~~> pool-name~~
  ~~> the pool-id associated with the resource.~~
  ~~> In case of clustered-pool, it can be a comma separated~~
  ~~> list of available connection pools.~~
  ~~>~~
  ~~643a650,656~~
  ~~> attributes~~
  ~~>~~
  ~~> pool-name~~
  ~~> the pool-id associated with the resource.~~
  ~~> In case of clustered-pool, it can be a comma separated~~
  ~~> list of available connection pools.~~
  ~~>~~

  ~~**Comments** : This is only a description change. Now pool-name can take comma-separated connection-pool ids. The DTD has no impact as such.~~

- DTD changes associated with introduction of new attributes and properties changed to attributes are available at http://www.glassfishwiki.org/gfwiki/attach/OnePagersOrFunctionalSpecs/DTD-Changes-Connection-Pool.html

  Comments : The new attributes for connection pool configuration. Also the properties which are changed to attributes now.

  Interfaces from AS 9.0 [Evolving-Uncommitted] that will be deprecated

  domain.resources.jdbc-connection-pools.test-pool.property.LazyConnectionAssociation=true
  domain.resources.jdbc-connection-pools.test-pool.property.LazyConnectionEnlistment=true
  domain.resources.jdbc-connection-pools.test-pool.property.AssociateWithThread=true
  domain.resources.jdbc-connection-pools.test-pool.property.MatchConnections=true

  AS 9.0 One Pager: http://appserver.sfbay.sun.com/as9ee/eng/connectors/connectionpool.txt

  Introduction of attributes in AS 9.1 corresponding to the above properties :

domain.resources.jdbc-connection-pools.test-pool.lazy-connection-association=true
domain.resources.jdbc-connection-pools.test-pool.lazy-connection-enlistment=true
domain.resources.jdbc-connection-pools.test-pool.associate-with-thread=true
domain.resources.jdbc-connection-pools.test-pool.match-connections=true

- System property : com.sun.enterprise.Connectors.SwitchoffACCConnectionPooling = true/false

4.5.2 Imported interfaces

4.5.3 Other interfaces (Optional)

4.6. Doc Impact

1. Administration Guide
2. Developer Guide
3. Reference
4. Admin GUI/CLI help

4.7. Admin/Config Impact

- ~~CLI commands for resource creation will accommodate connection-pool attributes and properties~~
    - ~~asadmin> create-jdbc-resource --user admin --passwordfile passwords.txt **--connectionpool=false --datasourceclassname datasource-classname --restype xa-datasource --property url=jdbc-url** jdbc/Resource~~

      ~~**Comments** : A jdbc resource is created without any connection pool. This is indicated by *connectionpool* being set to false. The required attributes/properties to connect to external database are provided at the jdbc resource creation itself. These attributes/properties are same as attributes/properties associated with *create-connection-pool* command. However connection-pool attributes like *steadypoolsize*, *maxpoolsize* should not be provided as connection pooling is switched off. In case those attributes are provided, an error must be thrown.~~
      ~~Internally a connection pool can be created and linked to jdbc resource to have no impact on DTD. The connection pool can have same name as resource-id, which in this case is *jdbc/Resource*. The attributes of connection pool can be set such that, it is evident that connection pooling is switched off. For example, in this case steadypoolsize, maxpoolsize, poolresize attributes of connection pool can be set to -1.~~

    - ~~asadmin> create-jdbc-resource --user admin --passwordfile passwords.txt **--connectionpool=true --datasourceclassname datasource-classname --restype xa-datasource --steadypoolsize 20 --maxpoolsize 100 --maxwait 120000 --poolresize 5 --property url=jdbc-url** jdbc/Resource~~

      ~~**Comments** : A jdbc resource is created with connection pool, as connectionpool attribute is set to true. The required connection pool attributes and properties are specified at the jdbc resource creation itself. These attributes/properties are same as attributes/properties associated with *create-connection-pool* command. Internally a connection pool is created and linked to jdbc resource. The connection pool can have same name as resource-id, which in this case is *jdbc/Resource*.~~

    - ~~asadmin> create-jdbc-resource --user admin --passwordfile passwords.txt **--connectionpool=true --connectionpoolid jdbc/ConnectionPool --datasourceclassname datasource-classname --restype xa-datasource --steadypoolsize 20 --maxpoolsize 100 --maxwait 120000 --poolresize 5 --property url=jdbc-url** jdbc/Resource~~

Comments : A jdbc resource is created with connection pool, as connectionpool attribute is set to true. The required connection pool attributes and properties are specified at the jdbc resource creation itself. These attributes/properties are same as attributes/properties associated with *create connection pool* command. Internally a connection pool is created with id *jdbc/ConnectionPool* and linked to jdbc resource. If a connection pool with same connection pool id already exists, an error must be thrown.

- Linking resource creation to connection pool creation in GUI. The resource creation page must be capable of taking required properties, to connect to external system, in case user does not need connection pooling. Also while creating resource, user must be able to create a new connection pool. The user experience on GUI must be same as CLI.

- Capability to select collection of connection pools at resource creation using GUI/CLI

  - asadmin> create-jdbc-resource –user admin –passwordfile passwords.txt ***–connectionpoolid node-1-pool, node-2-pool, node-3-pool*** jdbc/DerbyPool

    Comments : A jdbc resource is created with list of connection pools. This will enable pool clustering.

  - asadmin> create-connector-resource –user admin –passwordfile passwords.txt ***–connectionpoolid node-1-pool, node-2-pool, node-3-pool*** ConnectorPool

    Comments : A connector resource is created with list of connection pools. This will enable pool clustering.

- Segregation of additional connection pool properties into mandatory properties, sun-specific properties and driver-specific properties at the time of connection pool creation using GUI/CLI

  - asadmin> create-jdbc-connection-pool –user admin –passwordfile passwords.txt –datasourceclassname datasource-classname –restype xa-datasource –property url=jdbc-url ***–sun-specific-property LazyConnectionAssoc=true:LeakTracing=true:ConnectionLeakInterval=300:PollInterval=300*** jdbc/ConnectionPool

    Comments : The sun-specific properties are defined using attribute *sun-specific* instead of *property.*

- GUI/CLI support for new attributes
  - The new attributes will not be exposed in GUI/CLI at the time of connection pool creation. These attributes will be set to default values when connection pool is created.
  - In CLI user can use asadmin set command to changes the values of these attributes.
    - asadmin> set domain.resources.jdbc-connection-pool.*<jdbc-connection-pool-name>*.***connection-leak-timeout-in-seconds***=300
  - In GUI, the new attributes are available in a new tab named ***Advanced*** when user click on a connection pool. The user can edit values on that tab.

4.8. HA Impact

4.9. I18N/L10N Impact

4.10. Packaging & Delivery

4.11. Security Impact

4.12. Compatibility Impact

4.13. Dependencies

1. ~~Need Oracle RAC with 2-3 nodes for developing and testing this feature.~~

5. Reference Documents

1. CR 6436557 - Change request for Sun AppServer to provide load balancing for jdbc connections
2. CR 6436556 - Change request for Sun AppServer to support Oracle 10g Rac using Oracle 10g Type 4 jdbc thin driver
3. Oracle RAC documentation
4. CR 6472065 - JDBC ReclaimTimeout - Enhancement Request for JDBC Connection Pool management
5. CR 6473760 - RFE: Provide an option to physically close a connection after it has been acquired X times.

6. Schedule

6.1. Projected Availability

Aligned to overall Application Server 9.1 schedules