4

Connection Creation Consumer

Starting with Oracle Database Release 23ai, you can make UCP callback the caller application on every connection creation.

4.1 Implementing a Connection Creation Consumer

In the current release of Oracle Database, you can register a new connection creation consumer for a specific `PoolDataSource` object, which is called back on every database connection creation attempt, whether it is successful or unsuccessful.

Every connection creation operation triggers the acceptance of that consumer. For example, it can be triggered for a connection borrow request or it can be triggered with some UCP worker threads trying to keep up with a minimum pool size property, replacing a *bad* connection.

This section describes how to implement the connection creation consumer.

> **Note:**
>
> Once a registered consumer gets executed in a thread that creates a connection for the pool, you must implement it without blocking calls and intensive computations.

Example 4-1 Registering a Consumer

The following code snippet shows how to register a consumer:

```
import oracle.ucp.ConnectionCreationInformation;
import java.util.function.Consumer;
... rest of imports ...

  ...
  final PoolDataSource pds = new PoolDataSourceFactory.getPoolDataSource();
  ... configure pds ...
  final Consumer<ConnectionCreationInformation> consumer = p -> { ... };
  pds.registerConnectionCreationConsumer(consumer);
  ...
```

Once a consumer is registered, every connection creation operation triggers an accept call of a registered consumer.

Example 4-2 Unegistering a Consumer

The following code snippet shows how to unregister a consumer:

```
import oracle.ucp.ConnectionCreationInformation;
import java.util.function.Consumer;
... rest of imports ...

  ...
  final PoolDataSource pds = new PoolDataSourceFactory.getPoolDataSource();
  ... configure pds ...
  final Consumer<ConnectionCreationInformation> consumer = p -> { ... };
  pds.registerConnectionCreationConsumer(consumer);
  ...
  pds.unregisterConnectionCreationConsumer();
  ...
```

Once a consumer is unregistered, there are no more consumer acceptance calls for every connection creation request in a pool.

Example 4-3   Connection Creation Status Verification

The following code snippet shows how to verify whether a connection creation was successful or unsuccessful:

```
...
final Consumer<ConnectionCreationInformation> consumer = p -> {
  if (p.getStatus() == ConnectionCreationInformation.Status.SUCCESSFUL) {
    ... handle successful connection creation code path...
  }

  if (p.getStatus() == ConnectionCreationInformation.Status.FAILURE) {
    ... handle unsuccessful connection creation code path...
  }

  ... };
```

Example 4-4   Handling Failed Connection Creation

If the connection creation operation fails, then an accepted consumer throws the corresponding `SQLException` vendor error code. The following code snippet shows how to handle the error:

```
...
final Consumer<ConnectionCreationInformation> consumer = p -> {
  if (p.getStatus() == ConnectionCreationInformation.Status.FAILURE) {
    final int errorCode = p.getErrorCode();
    ... do some error code path...
  }

  ... };
```

Example 4-5   Retrieving Information About Successful Connections

The following code snippet shows how to retrieve information about successfully created connections:

```
...
final Consumer<ConnectionCreationInformation> consumer = p -> {
  if (p.getStatus() == ConnectionCreationInformation.Status.SUCCESSFUL) {
    // Net Connection ID for an appropriate connection.
    final String netConnId = getNetConnectionId();

    // database instance (if applicable) name, on which connection was 
created.
    final String instanceName = getInstanceName();

    // database service (if applicable) name, on which connection was created.
    final String serviceName = getServiceName();

    // database host (if applicable) name, on which connection was created.
    final String hostName = getHostName();

    // database unique ID for created connection.
    final String dnUniqId = getDatabaseUniqId();

    // database instance (if applicable) unique ID for created connection.
    final String instanceId = getInstanceId();

    // Security information for created connection.
    final SecurityInformation securityInfo = getSecurityInformation();

    ... rest of successful connection creation code path ...
  }
  ... };
```