Table of Contents

CORAL Connection Strings

CORAL supports two types of connection strings, those pointing explicitly to physical connections, as well as CORAL aliases resolved at runtime using the replica lookup functionality. For instance:

- `oracle://oraserver1/conditiondataowner` (connect to the Oracle server with TNS identifier `oraserver1` to use the schema called `conditiondataowner`)
- `mysql://myserver1/conditiondatabase` (connect to the MySQL server on host `myserver1` to use the MySQL database `conditiondatabase`)
- `sqlite_file:conditiondata.db` (connect to the SQLite file identified by relative path `conditiondata.db`)
- `sqlite_file:/tmp/conditiondata.db` (connect to the SQLite file identified by absolute path `/tmp/conditiondata.db`)
- `MyConditionDatabase` (connect to one of the physical replicas corresponding to the logical alias `MyConditionDatabase`)

CORAL database lookup

Logical aliases are resolved at runtime by CORAL to physical connection strings using one of the implementations of the lookup service interface. If the default XML lookup service is used, you must have a `dblookup.xml` file describing the `MyCoolDatabase`, for instance:

```
<?xml version="1.0" ?>
<servicelist>
  <logicalservice name="MyConditionDatabase">
    <service name="mysql://myserver1/conditiondatabase" accessMode="readonly" authentication="pas
    <service name="oracle://oraserver1/conditiondataowner" accessMode="update" authentication="pa
  </logicalservice>
</servicelist>
```

In this case, assuming that you are connecting to CORAL or COOL in read-only mode, CORAL will first try to connect to the MySQL replica on `myserver1` (and may then retry to connect to the Oracle replica on server `oraserver1` if the MySQL connection failed).

CORAL authentication

The main authentication mechanism in CORAL is based on usernames and passwords. If the default XML authentication service is used, you must have an `authentication.xml` file listing the username/password pairs which you want to be authenticated on each physical connection. CORAL also includes the concept of "role", which is an abstraction of the concept of user. For the same physical connection string, several username/password pairs can be declared to the authentication service for different CORAL "roles". By selecting a role, users are able to select under which username they want to connect to the database. Roles are selected in CORAL through the C++ API; in COOL, they can be selected in the COOL connection string, as described further below. The `authentication.xml` file could look like the following, for instance:

```
<?xml version="1.0" ?>
<connectionlist>
  <connection name="oracle://ora_srv/cool_main">
    <parameter name="user" value="dafault_user" />
    <parameter name="password" value="PASSWORD" />
    <role name="writer">
      <parameter name="user" value="oracle_writer" />
      <parameter name="password" value="PASSWORD" />
    </role>
    <role name="owner">
      <parameter name="user" value="cool_main" />
      <parameter name="password" value="PASSWORD" />
```

```
    </role>
  </connection>
  <connection name="mysql://myserver1/cooldb">
    <parameter name="user" value="mysql_reader" />
    <parameter name="password" value="PASSWORD" />
    <role name="writer">
      <parameter name="user" value="mysql_writer" />
      <parameter name="password" value="PASSWORD" />
    </role>
  </connection>
</connectionlist>
```

CORAL (and COOL) also support Kerberos authentication to Oracle as an alternative to username/password authentication (see task #30593). Kerberos authentication must be explicitly enabled on your database server (by default it is not). To use Kerberos authentication, a null password must be specified in the CORAL `authentication.xml` file: if the username is also null, then you will connect to the account with the same name as your Kerberos credentials (e.g. `avalassi@cern.ch`); if a username is given, you will connect as that Oracle user via proxy authentication with your Kerberos credentials. Kerberos authentication in CORAL and COOL is still experimental - use at your own risk! Kerberos non-proxy authentication will NOT be fully supported in COOL (e.g. in the coolAuthentication package) because its production use is highly unlikely. Only Kerberos proxy authentication will be slowly integrated into COOL.

For instance:

```
<?xml version="1.0" ?>
<connectionlist>
  <connection name="oracle://lcg_cool_nightly/lcg_cool_nightly">
    <!-- KERBEROS PROXY -->
    <parameter name="user" value="lcg_cool_nightly" />
    <parameter name="password" value="" />
  </connection>
  <connection name="oracle://lcg_cool_nightly/avalassi@cern.ch">
    <!-- KERBEROS USER -->
    <parameter name="user" value="" />
    <parameter name="password" value="" />
  </connection>
</connectionlist>
```

COOL Connection Strings

To connect to a COOL database, the user must pass to the COOL API the details for the connection, in the form of a connection string.

One COOL 'database' is a self-consistent and fully self-contained set of tables within the schema owned by a given Oracle account. The names of all such tables start with a given prefix, the COOL 'database name' (which must be uppercae and up to 8 characters long).

COOL connection string formats

The COOL software understands two formats for a connection string:

1. *CORAL-aware COOL connection string*:
   `<CORAL connection string>[(<CORAL role>)]/<COOL database name>`
2. *Old-style COOL connection string with explicit username and password*:
   `<engine>://<server>;schema=<schema>;dbname=<COOL database name>;[user=<user>;password=<password>]`

The first format allows users to take advantage of CORAL advanced functionalities (including replica lookup with automatic fallback, user authentication and authorization with different roles, Kerbers authentication). *The CORAL-aware COOL connection string is the only supported format for production use.*

The second format (old-style COOL connection string with explicit username and password) is provided only for backward compatibility and for quick tests. It has several well-known limitations which may lead to unexpected behaviour (in particular, see bug #69129 and Richard's comments about multiple COOL connections in his ATLAS twiki). *The old-style COOL connection string with explicit username and password is NOT supported for production use.*

1. CORAL-aware COOL connection string

The connection string `<CORAL connection string>[(<CORAL role>)]/<COOL database name>` is made of three parts. The first and the third are compulsory, while the second is optional.

CORAL connection string

The first part of the COOL connection string, `<CORAL connection string>`, is passed directly to CORAL. Both types of connection string supported by CORAL are allowed, those pointing explicitly to physical connections, as well as CORAL aliases resolved at runtime using the replica lookup functionality. The valid syntax for CORAL connection strings is described above in this page.

CORAL role

The second part of the COOL connection string, `(<CORAL role>)`, is also CORAL specific. This is optional in COOL and, if it exists, it must be surrounded by parentheses. It declares the CORAL role to use for the COOL connection. CORAL roles and their definition in the `authentication.xml` file are described above in this page.

Using the `authentication.xml` and `dblookup.xml` example files above in this page, if you do not specify the role, you will use the user default_user when connecting to the Oracle database, while mysql_reader will be used to connect to MySQL. If you specify the role to be writer with something like MyCOOL(writer), you will use oracle_writer and mysql_writer respectively for Oracle and MySQL.

COOL database name

The third part of the connection string must be separated from the first two parts by a "/" and can consist of up to 8 upper case alphabetic characters, numbers or "_" (underscore) (the equivalent regular expression is "[A-Z][A-Z0-9_]{0,7}"). Valid examples are `COOLDB`, `MARCO123`, `TEST_N1`. This string will identify the COOL database instance inside an Oracle or MySQL schema or inside an SQLite file.

Summary about CORAL-aware COOL connection strings

Putting all together, these are a few examples of valid connection strings:

- `oracle://ora_srv/cool_main(owner)/TEST_1`
- `mysql://myserver1/cooldb/COOLDB`
- `sqlite_file:mydir/MyDB.db/MARCO123`
- `MyCOOL/COOLDB`
- `MyCOOL(writer)/TEST_1`
- `oracle://lcg_cool_nightly/avalassi@cern.ch/COOLTEST`

2. Old-style COOL connection string with explicit username and password

The old style connection string is `<engine>://<server>;schema=<schema>;dbname=<COOL database name>;[user=<user>;password=<password>]`. As already said, this connection string format is mainly for backward compatibility and can be replaced with an equivalent CORAL-aware connection string.

The old style connection string `<engine>://<server>;schema=<schema>;dbname=COOLDB` is equivalent to the CORAL-aware one `<engine>://<server>/<schema>/COOLDB`. The only thing that is not available with the new format is the explicit specification of username and password. Note: to make the old-style connection string to work with username and password, both have to be specified otherwise they will be ignored.

For any questions please contact MarcoClemencic or AndreaValassi.

-- AndreaValassi - 05-Jun-2013

This topic: Persistency > CoolConnectionStrings
Topic revision: r4 - 2013-06-05 - unknown