# Oracle® Enterprise Data Quality

Integrated Version Control

Release 11g R1 (11.1.1.7)

**E40041-01**

October 2013

Oracle Enterprise Data Quality (EDQ) supports integration with Subversion, which is a version control system. This document explains how the integration works and how it is expected to be used.

> **Note:** EDQ currently only supports integration with Subversion 1.6 and 1.7. Attempting to integrate with a more recent version will cause an error.

This document is intended for advanced users of EDQ and administrators responsible for integrating EDQ with third-party applications.

## 1 Prerequisites

The Subversion server with which EDQ is being integrated must meet these prerequisites:

- It must support HTTP/DAV access.
- It must require authentication on commit.
- It must not require authentication on checkout or update.

## 2 Restrictions and Limitations

When Subversion is integrated with EDQ as a store of configuration information, the following restrictions and limitations apply. Consider these items before deciding to configure integrated version control using Subversion.

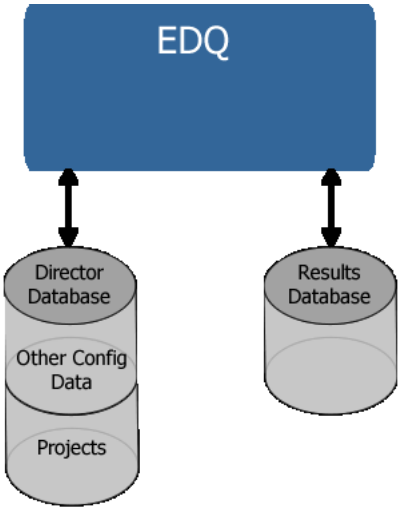
- You cannot update or revert an item that is open in the Director interface or the Subversion server.
- You cannot rename a project once the project is under version control.
- Deleting a project does not remove it from the Subversion repository.
- Case-insensitive name matching is used.

## 3 Integration Architecture

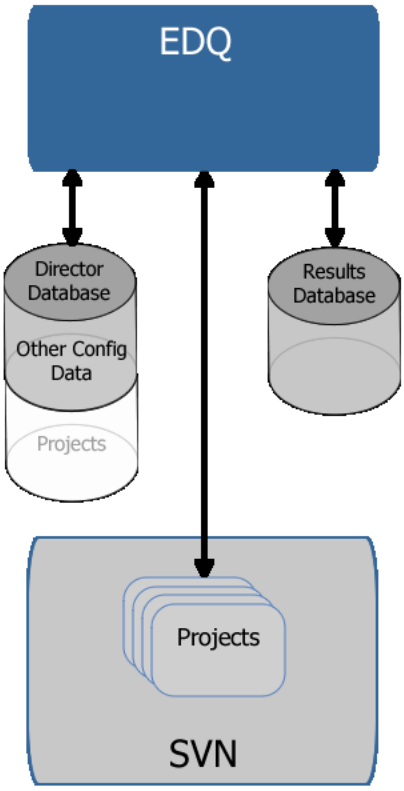
The EDQ server can be configured to be aware of a Subversion server as a store of configuration information.

**Note:** In this instance, configuration information means information that is managed using the Director UI; for example, projects and system-level data.

In a standard EDQ instance, configuration information, including project information, is stored in the Director database:




The following figure shows an EDQ instance integrated with Subversion:



**Note:** The Director database is still required because it contains data derived from the file-mastered configuration that has been normalized to allow querying by the applications.

With EDQ configuration files mastered and stored in a Subversion repository, a Subversion client can be used to commit or otherwise access them. Because EDQ includes an embedded Subversion client, Subversion client operations to control configuration changes can be performed directly in Director once the EDQ integration with Subversion has been enabled.

# 4 Setting Up a Repository

The first stage of configuration is to create a workspace directory where the checked out data will be stored:

1. Create a directory on the disk where desired (for example, `C:\MyRepository`) and then add it and commit it to Subversion.
2. Inside the newly created directory, set the following Subversion properties:

```
svn propset svn:ignore .metadata .
svn propset edq:systemversion 11.1.1:base .
```

3. Commit these changes into Subversion. Your workspace now displays these properties:

```
svn proplist -v .
Properties on '.':
  svn:ignore
    .metadata
  edq:systemversion
    11.1.1:base
```

4. Create the following subdirectories in the newly created directory:
   - Data Stores
   - Hidden Reference Data
   - Images
   - Projects
   - Published Processors
   - Reference Data
5. Add and commit these directories. The repository is now set up correctly for EDQ.

The preceding steps only need to be performed once per repository. All remaining changes can be made using EDQ.

# 5 Configuring EDQ

Subversion must be integrated with a fresh installation of EDQ.

**Caution:** When an EDQ instance is integrated with Subversion, all pre-existing and other configuration information is lost. To retain this information, you must package and export it first. For further details, see Section 5.2, "Retaining Existing Configuration Information."

**Note:** Oracle recommends that a single workspace be assigned to each instance of EDQ because it is difficult to move between workspaces in a single EDQ instance.

## 5.1 Configuring a New EDQ Installation

To configure a new EDQ installation:

1. Shut down the application server.
2. Check-out the workspace from Subversion. It is not necessary to checkout the whole tree; just the workspace directory itself is required.
3. Add the following line to the `config/director.properties` file, replacing the directory path with that of the absolute path to the root workspace directory; for example:

   ```
   sccs.workspace = C\:\\MyRepository
   ```

   **Note:** This example demonstrates the need to escape colon (:) and backslash (\) characters in the path with a backslash. You must also escape space characters in the path with a backslash.

4. Start the application server, and start Director.
5. Check the top of the `Main0.log` file for an INFO message listing the name of the SCCS workspace; for example:

   ```
   INFO: 02-Sep-2013 10:05:21: SCCS workspace is C:\MyRepository
   ```

6. If no errors follow this message, EDQ is configured to use Subversion. If there are errors, see Section 8, "Troubleshooting," for possible solutions.

## 5.2 Retaining Existing Configuration Information

As previously stated, Subversion must be integrated with a fresh installation of EDQ.

Therefore, any pre-existing projects and other configuration items in an EDQ installation must be packaged before integration begins and then imported to the new installation afterwards:

1. Package all configuration items in the current EDQ instance into DXI files.
2. Install a new instance of EDQ with the Subversion integration enabled.
3. Import the DXI files into the new instance, and commit the files to the Subversion workspace.
4. Check that the configuration items are all valid and working correctly.

Note that all passwords for Data Stores must be re-entered after a configuration import.

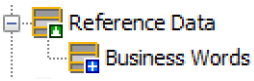
5. Decommission the previous instance.

# 6 Version Control Interface

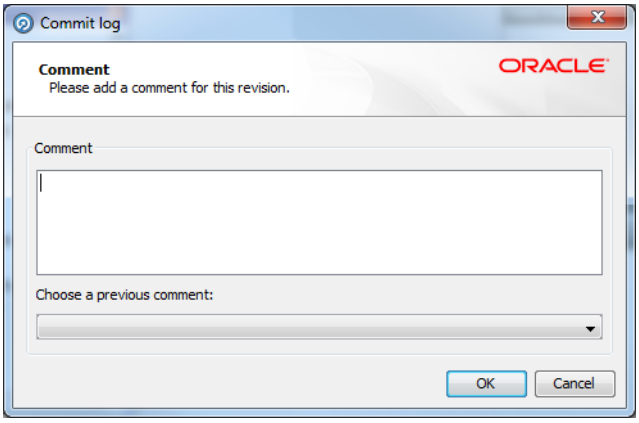
Once EDQ is integrated with Subversion enabled, the following interface elements become visible within the Director application:

- Subversion status icon overlays in Project Browser - There are two icons used to indicate the three possible Subversion statuses of nodes in the Project Browser:
  - No icon - The node (and its sub-nodes) are all up to date.
  -  - This node (and its sub-nodes) have modifications.
  -  - This node (and its sub-nodes) is new/currently not under Version Control.

  For example, the following image shows both icons in use. The Reference Data node is modified (green icon) as one of its sub-nodes has changed. A new piece of Reference Data - Business Words - has been added, and is marked with the blue icon:

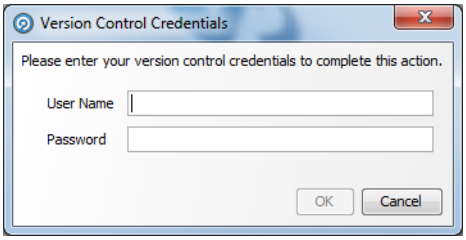



- Version Control tab - The Properties dialog (displayed by right-clicking on an item in the Project Browser and selecting Properties) now contains a Version Control tab that describes the state of the item: when it was last updated, its Subversion revision, whether it's up-to-date, and so on.
- New context menu for Version Control - The Project Browser right-click menu now contains a Version Control option. When selected, this displays a sub-menu with Subversion options to update, commit, revert, compare or view the log for the item. These options are recursive; for example, if you perform View Log on a single process then you will see the log for this process only, but if you perform View Log on the Processes node you will see changes for all processes.
- Comment and credentials dialogs on commit - When you commit changes to the repository, Director displays the Commit log dialog:




In this dialog you can enter a comment describing the change. Instead of entering a comment in the Comment box, you can autofill the box by choosing a comment from the list of comments previously entered in the current session.

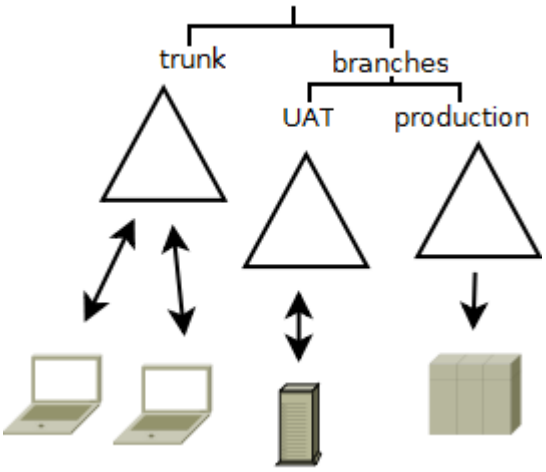
After you click OK in the Commit log dialog, Director displays the Version Control Credentials dialog if you have not already provided your credentials in the current session:




In this dialog you enter your user name and password for the Subversion repository and then click OK.

# 7 Deployment Example

An example deployment is presented here. In this illustration, there is a single Subversion server that holds three copies of the configuration for four EDQ installations:




The copies of the configuration are:

- **trunk** - the traditional location that all development work is performed on. New features of the configuration are developed and saved here.
- **branches/UAT** - this branch represents the copy of the configuration under UAT testing.
- **branches/production** - this branch represents the production copy of the configuration.

The four EDQ installations using the Subversion server for storing their configuration are:

- Two development laptops where design work and maintenance of existing projects are carried out.
- A UAT server for User Acceptance Testing changes.

- A production server for production runs.

In this example deployment, the laptop users develop configuration for individual projects on their own laptops and then commit changes back to the subversion repository on "trunk". Where the developers are co-operating on developing a project they will periodically update their local installation to pick up changes from the other developer.

At some point development reaches a point where it needs to be released to UAT for testing. A release manager then copies the necessary projects from "trunk" to "UAT" on the subversion server.

For example, the following Subversion command may be used:

```
svn cp -m"Release Project X to UAT" http://svn/repos/config/trunk/ProjectX
http://svn/repos/config/branches/UAT
```

The test manager then updates the UAT server's projects to load the new configuration into the OEDQ server. Over a period of time testing continues. As issues are found they are fixed in the UAT environment and committed back to the subversion repository.

Once UAT environment has achieved an acceptable test level it is promoted to release. This achieved in much the same way as the release from development to UAT. The necessary projects are copied across in the version control repository and then the production server is updated to use this configuration.

# 8 Troubleshooting

| Error | Cause and Solution |
|---|---|
| `Configuration database is not compatible with workspace` | The database has been used with a different workspace. This error usually arises occurs when operations have been performed in EDQ before Subversion version control is enabled.There are two solutions: drop/recreate the Director database, or reinstall EDQ. |
| `Unable to open an ra_local session to URL` | This may occur when trying to commit files to an invalid repository. The EDQ integration is not compatible with file-based repositories (repositories beginning with `file:///` or `C:\example`). A fully-declared http:// path to the repository must be made. |

# 9 Related Documents

For more information, see the following documents in the Oracle Enterprise Data Quality documentation set:

- *Oracle Enterprise Data Quality Release Notes*
- *Oracle Enterprise Data Quality Architecture Guide*

See the latest version of this and all documents in the Oracle Enterprise Data Quality Documentation website at

http://download.oracle.com/docs/cd/E48549_01/index.htm

# 10 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

### Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

---

Oracle Enterprise Data Quality Integrated Version Control, Release 11g R1 (11.1.1.7)
E40041-01