Getting started with Team Coding

Toad for Oracle

Quest

Table of Contents

Quest

About Team Coding

Team Coding provides a convenient way to control database objects and scripts within a team environment through a single common interface. You can also optionally link Team Coding to an external Version Control System (VCS) in order to maintain revision history for changes to your database objects and scripts in your team environment.

Unlike other IDEs which support version control, Team Coding enables developers to work directly on database objects such as PL/SQL code and ensures the corresponding files in version control are kept in sync. If a versioned object is changed outside of Toad, Team Coding will report an out-of-sync and enable the developer to take the appropriate course of action.

All source control is managed through the Team Coding menu in Toad.

Working with a Version Control System

Team Coding can be linked to an existing VCS provider to manage full source control and revision history capabilities. You can check objects into and out of the VCS to work on them in the Editor or Schema Browser. Team Coding keeps track of changes to objects and prevents modifications to the code in the database unless a Toad user checks the source out of the VCS through Team Coding.

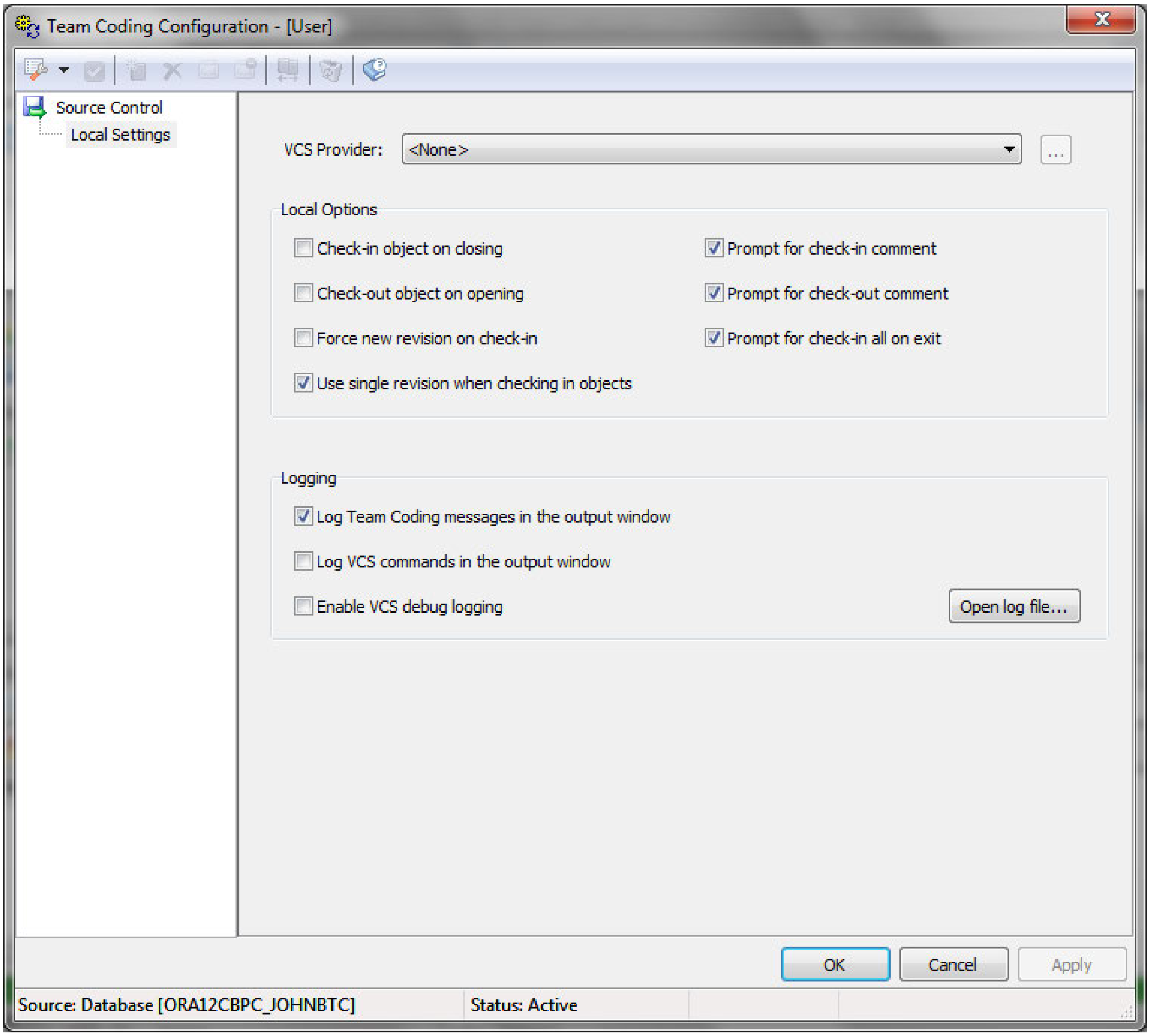
Working without a Version Control System

Team Coding without a link to a VCS provider enables your team to manage simultaneous changes to your source code in the database, but it will not provide revision history for those changes. If you choose to use Team Coding in this fashion:

- Administrators must protect production schemas by creating one or more Team Projects, which will be the only way that developers can check objects in and out of the database and work on them.
- When a developer checks out an object through a Team Coding Project, Toad locks the object in the database. This prevents other developers from unintentionally changing the object source before the current developer has a chance to apply his or her changes. When finished making the necessary changes, the developer then checks the finished code into the database through Team Coding, which unlocks the object again for editing by other developers. (This follows the standard lock-modify-unlock method for source control).
- Team Coding will not maintain a revision history as using it with a VCS normally would. Access control to the object is maintained in the database, but no versions of the source are available other than the current revision.

Setting up Team Coding

To set up Team Coding in Toad, connect to the database which you want to control through Team Coding and select ***Configuration…*** from the ***Team Coding*** main menu. You will be presented with the following screen:



Installing Team Coding

If this is your first time installing Team Coding, you will need to click on the Team Coding installation button and select ***Install Team Coding to database…*** The installation wizard will guide you through the installation process and will allow you to install the Team Coding objects into your own schema or the centralized TOAD schema, which is the default recommended option.

Once the installation completes, you'll be ready to configure your Team Coding settings.

Upgrading Team Coding

If a previous version of Team Coding from Toad 12.6 or earlier is already installed on the database, the options under the Team Coding installation button will change to ***Upgrade Team Coding in database…*** Team Coding must be upgraded from its earlier version in order to be used in the current version of Toad. The installation wizard will guide you through the installation process and will allow you to install the Team Coding objects into your own schema or the centralized TOAD schema, which is the default recommended option. It will also provide an option to automatically migrate your previous Team Coding settings to this version of Team Coding.

Once the upgrade completes, you'll be ready to configure your Team Coding settings.

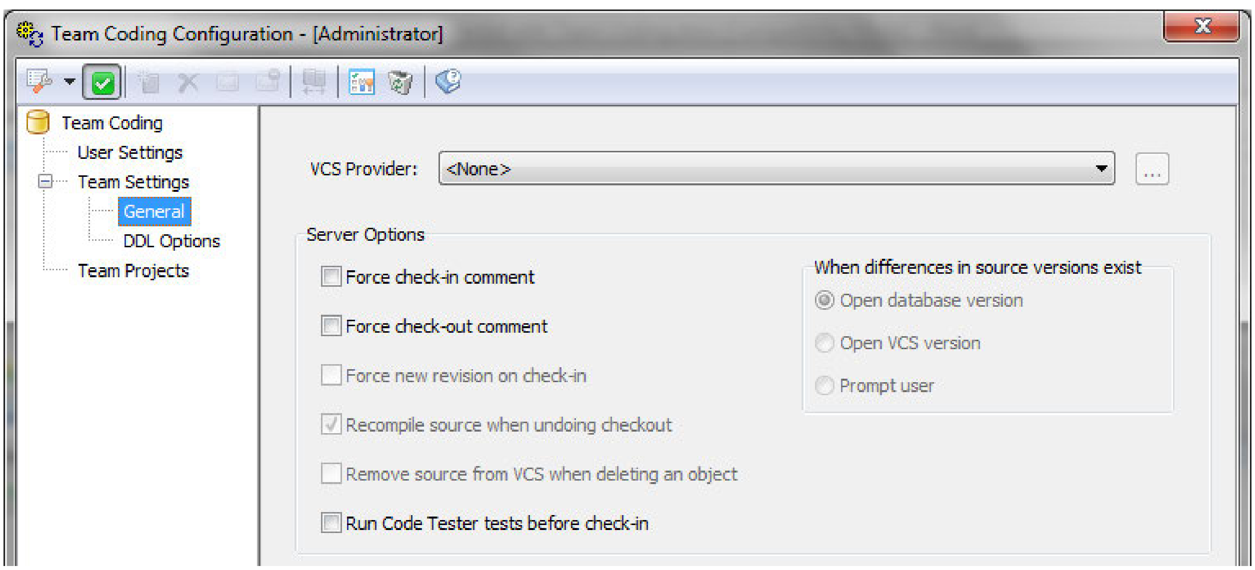
Setting up a Version Control Provider

Team Coding can be linked to an existing VCS Provider to manage full source control and revision history capabilities. Using Team Coding with a version control provider is highly recommended because it will allow you to maintain full revision history for your database object source in case a change to an object's source inadvertently becomes a "breaking" change. Without a link to an external version control system (VCS), you would have no way to restore the object's source to a previous version.

Team Coding currently supports the following Version Control Providers:

- CVS - Concurrent Versions System
- Git
- IBM Rational ClearCase
- Microsoft Team Foundation Server
- Microsoft Visual Studio Team Services
- Perforce
- Serena PVCS Version Manager
- Subversion

To link to an external Version Control provider, select the ***General*** node underneath ***Team Settings*** from the tree view on the left and select the corresponding provider from the ***VCS Provider*** drop-down box:



To set provider-specific options for the VCS Provider, click the ellipses button next to the version control provider drop-down.

Quest

Setting Team Options for Users

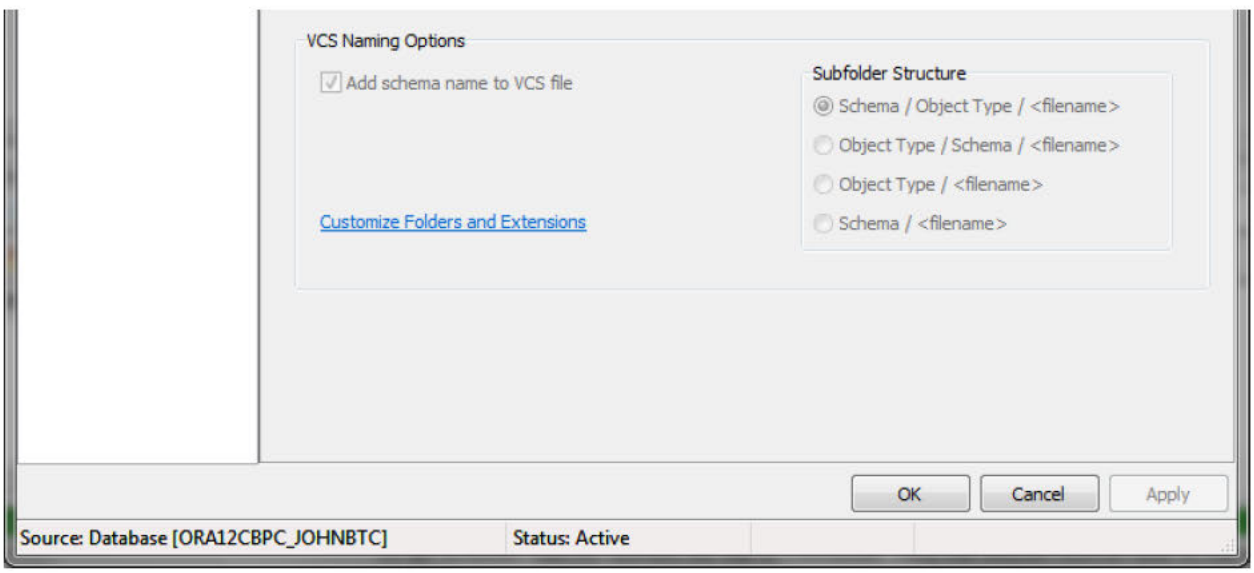
Once you select a Version Control provider, other options will enable which will allow you to specify how your team will work with Team Coding and the Version Control System. In the ***Server Options*** box, you can specify:

Force check-in comment	This option forces users to provide a comment when checking in objects to Team Coding. Users will not be able to check in the object if a comment is not provided. Comments will also be saved in the VCS if the VCS provider supports comments upon check-in.
Force check-out comment	This option forces users to provide a comment when checking out objects in Team Coding. Users will not be able to check out the object if a comment is not provided. Comments will also be saved in the VCS if the VCS provider supports comments upon checkout.
Force new revision on check-in	This option will force a new revision whenever a check-in happens – even if no modifications were made to the object's source. By default, Toad will recognize when objects haven't been changed and will not create a new revision unless this option is checked.
Recompile source when undoing checkout	This option will automatically recompile the previous source from the VCS provider into the database when an undo checkout is performed.
Remove source from VCS when deleting an object	This option allows the object to be automatically removed from the VCS provider when it is deleted from Team Coding. Note: Use this option with extreme caution as removing objects from the VCS also removes the object's entire revision history.
Run Code Tester tests before check-in	If the Code Tester database is installed, this option will run any predefined Code Tester test for an object before checking the object into the VCS provider. If the tests fail, the check-in process will also fail.
When differences in source versions exist	This option allows you to specify the default action when differences exist between the source in the VCS and the source in the database. "Prompt" is the default action, which will prompt the user for the desired action to take.

Customizing How Objects are Stored in the VCS Provider

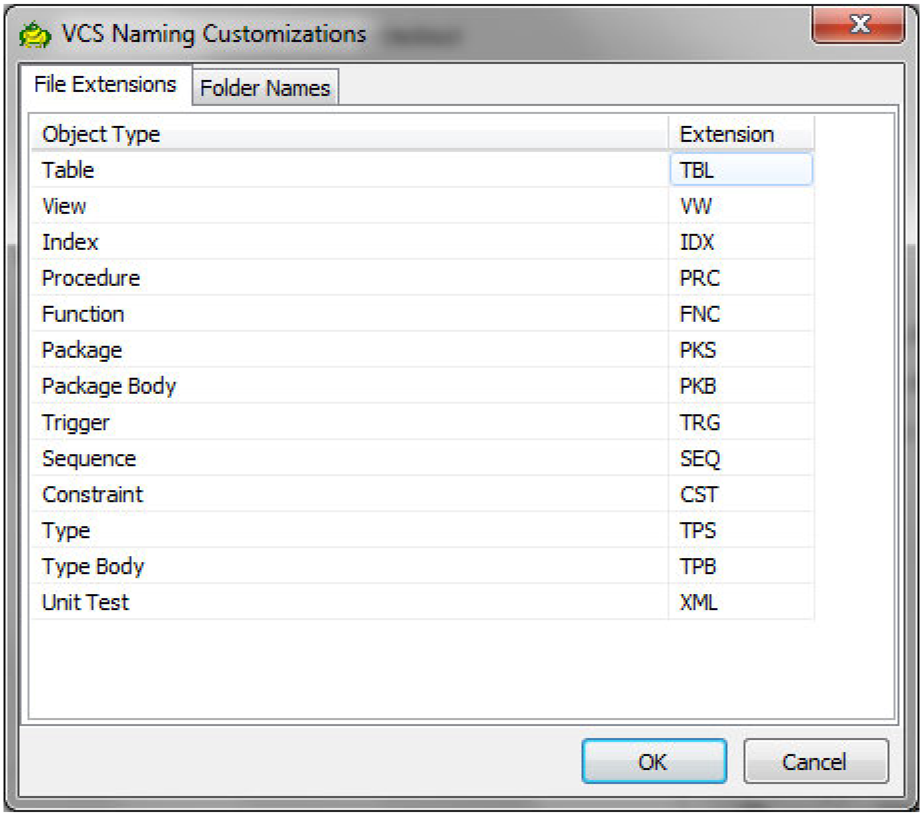
You can customize how your objects will be stored in the version control system. In the VCS Naming Options box, you can specify:

Add schema name to VCS file	This option allows you to specify whether the schema is prepended to the object's filename when stored in the VCS. When selected, objects in the VCS will be stored as: <Schema>.<Object Name>.<Extension> By default, this option is selected.
Subfolder Structure	This option allows you to define how objects are organized in the VCS when using subfolders (see next section). You can specify to structure your VCS subfolders using any of the provided folder structures. By default, "Schema / Object Type / <filename>" is selected.



Advanced VCS Naming Options

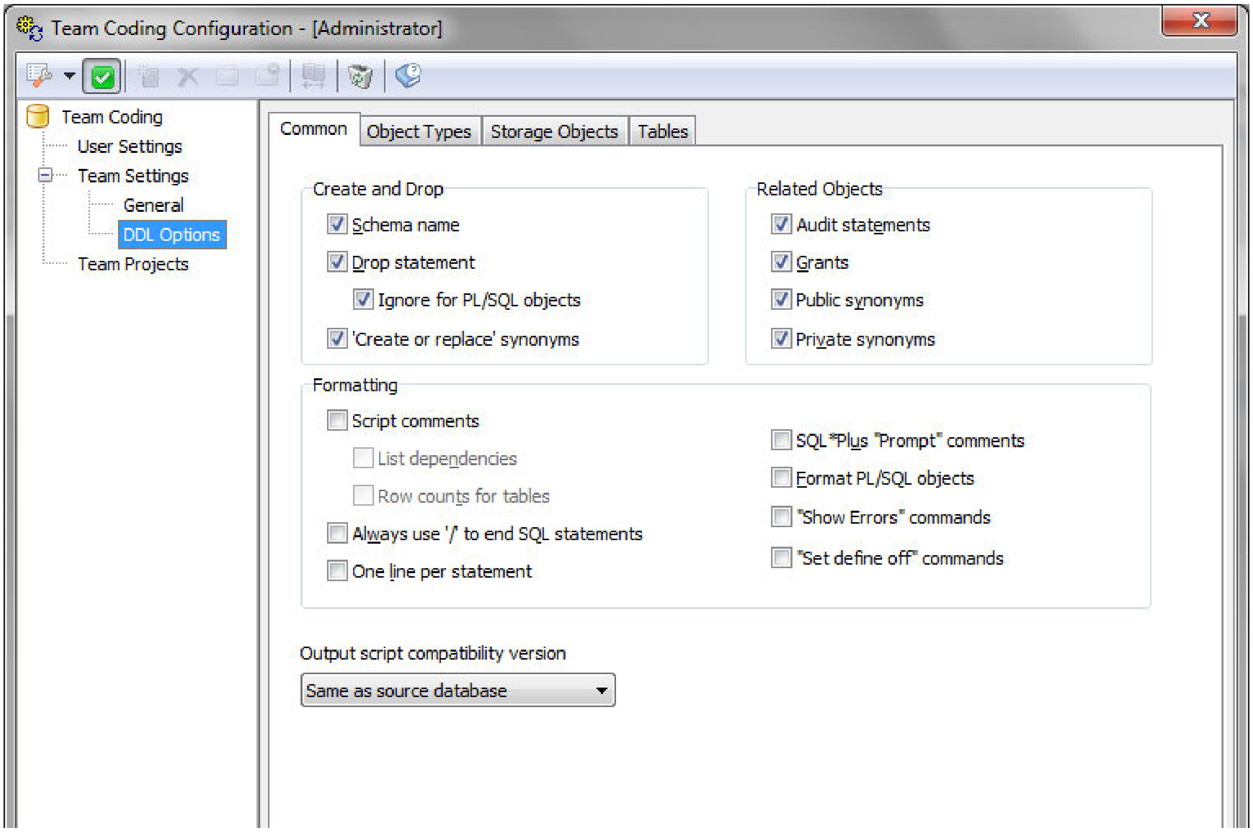
You can further customize how objects are stored in the VCS by clicking the ***Customize Folders and Extensions*** link in the VCS Naming Options box. In this dialog, you can customize the folder names and file extensions for each object type stored in the VCS.



By default, Toad will use standard file extensions and folder names for its object types, but you can change them here if you so desire.

Setting DDL Options

You can control how object DDL is extracted from the database in order to be saved to the VCS provider. To do this, select the ***DDL Options*** node underneath ***Team Settings*** from the tree view on the left.



This page provides options similar to the Export DDL functionality within Toad; however, these options are applied to all objects controlled by Team Coding when extracting the DDL to save to the VCS provider, regardless of local export settings defined by the user.

Note: It is also possible to extract the data from tables, and include INSERT statements into the exported DDL script using these options. This can result in very large scripts to be saved to the VCS and can significantly affect the performance of Team Coding. As a result, this option should only be used with extreme caution.

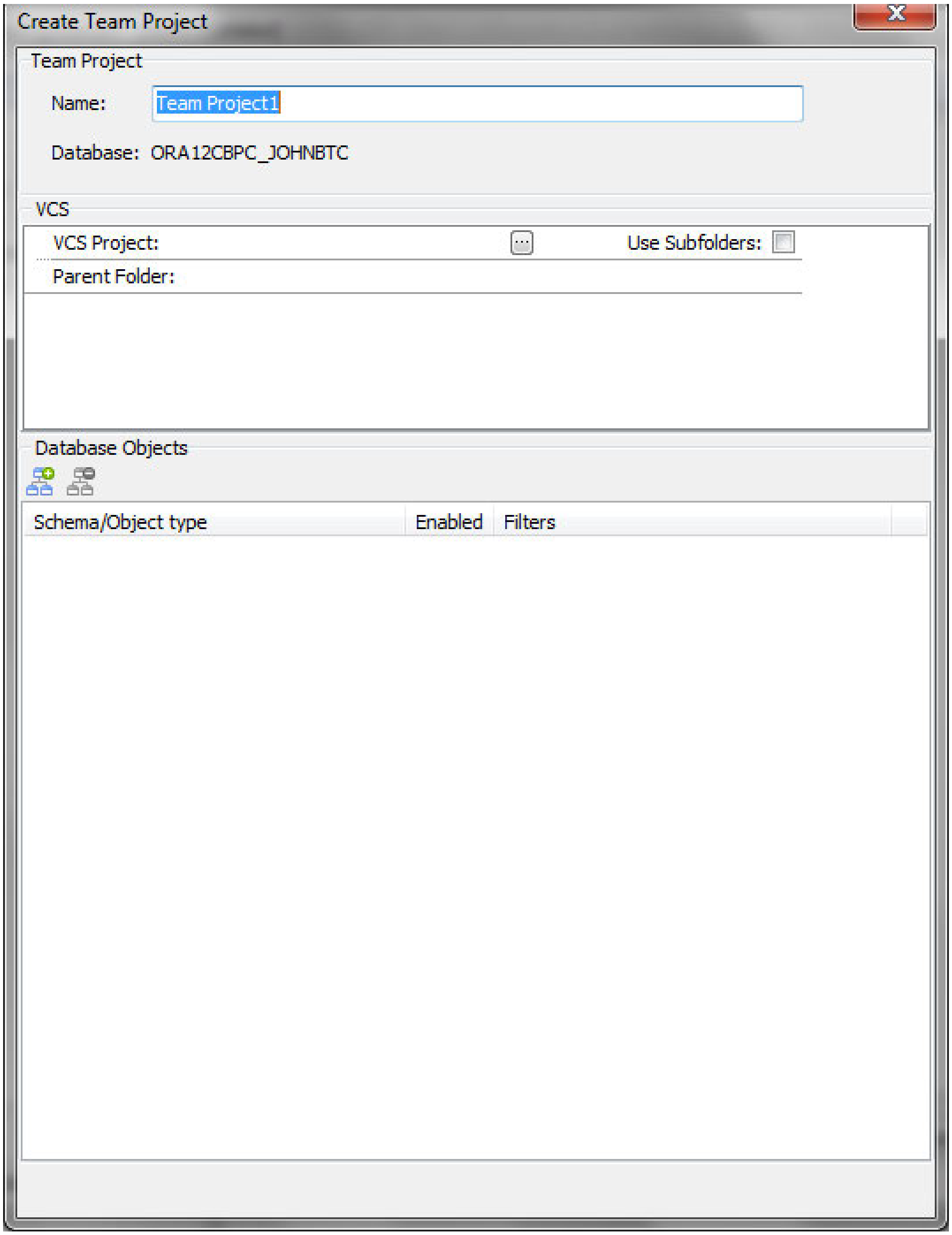
Setting up Team Projects

Team Projects allow you to define what database objects will be controlled by Team Coding. By default, Team Coding can control the following object types:

- Table DDL
- Views
- Indexes
- Procedures
- Functions
- Package Specifications
- Package Bodies
- Triggers
- Sequences
- Constraints
- Type Specifications
- Type Bodies

You can define which object types you want to control and then place filters on those object types to fine-tune which objects will be controlled by Team Coding.

To set up a Team Project, select the ***Team Projects*** node from the tree view on the left and click on the ***Create a new Team Project*** tool button . This will open the following window:



Selecting a VCS Project

Click on the ellipses button next to ***VCS Project*** to select the project in the version control system where the objects controlled by this Team Project will be stored.

Select the ***Use Subfolders*** option if you wish to organize the database objects in subfolders based off the ***Subfolder Structure*** option previously selected.

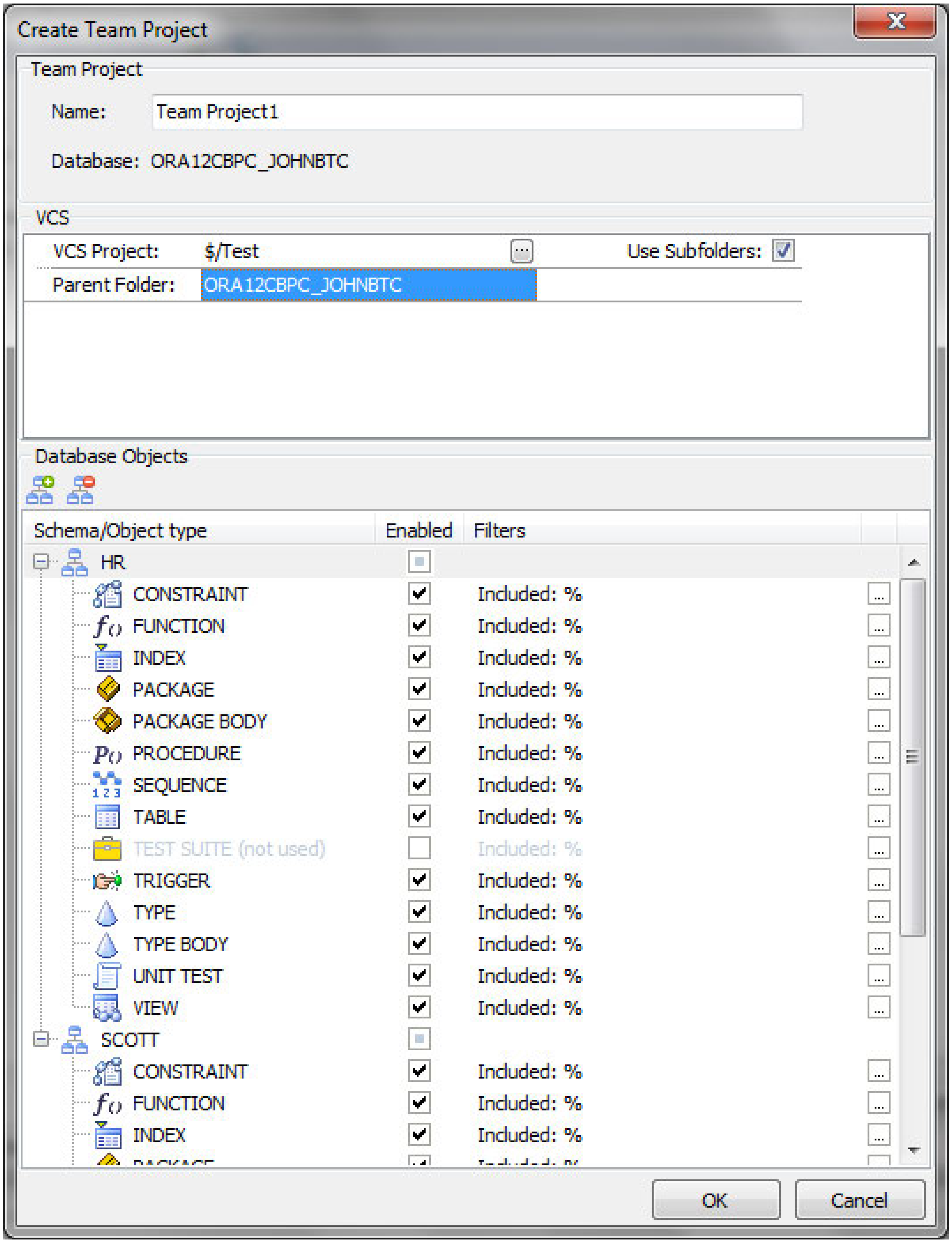
Note: If this option is not selected, all database objects will be placed in the folder defined in the ***VCS Project*** field.

The database name will automatically be entered into the ***Parent Folder*** field; however, this can be changed to whatever you prefer. It can also be left blank if you don't want to use a parent folder, but want the subfolders to be created directly underneath the ***VCS Project*** folder.

Selecting Database Objects

After specifying the VCS project, click the ***Add a Schema*** tool button to select which schemas will be controlled by Team Coding. A list of schemas within the database will be presented. You can either select multiple individual schemas to control, or select the "All Schemas" option, which will control all schemas in your database.

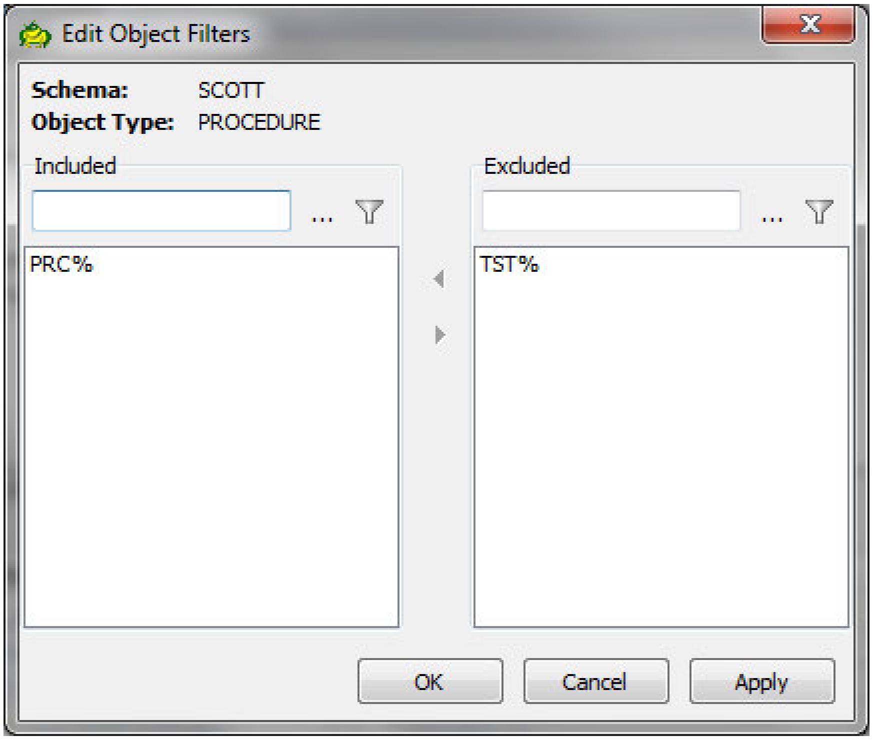
Once you have selected your schemas and press OK, the schemas and their object types will be listed in the ***Database Objects*** window:



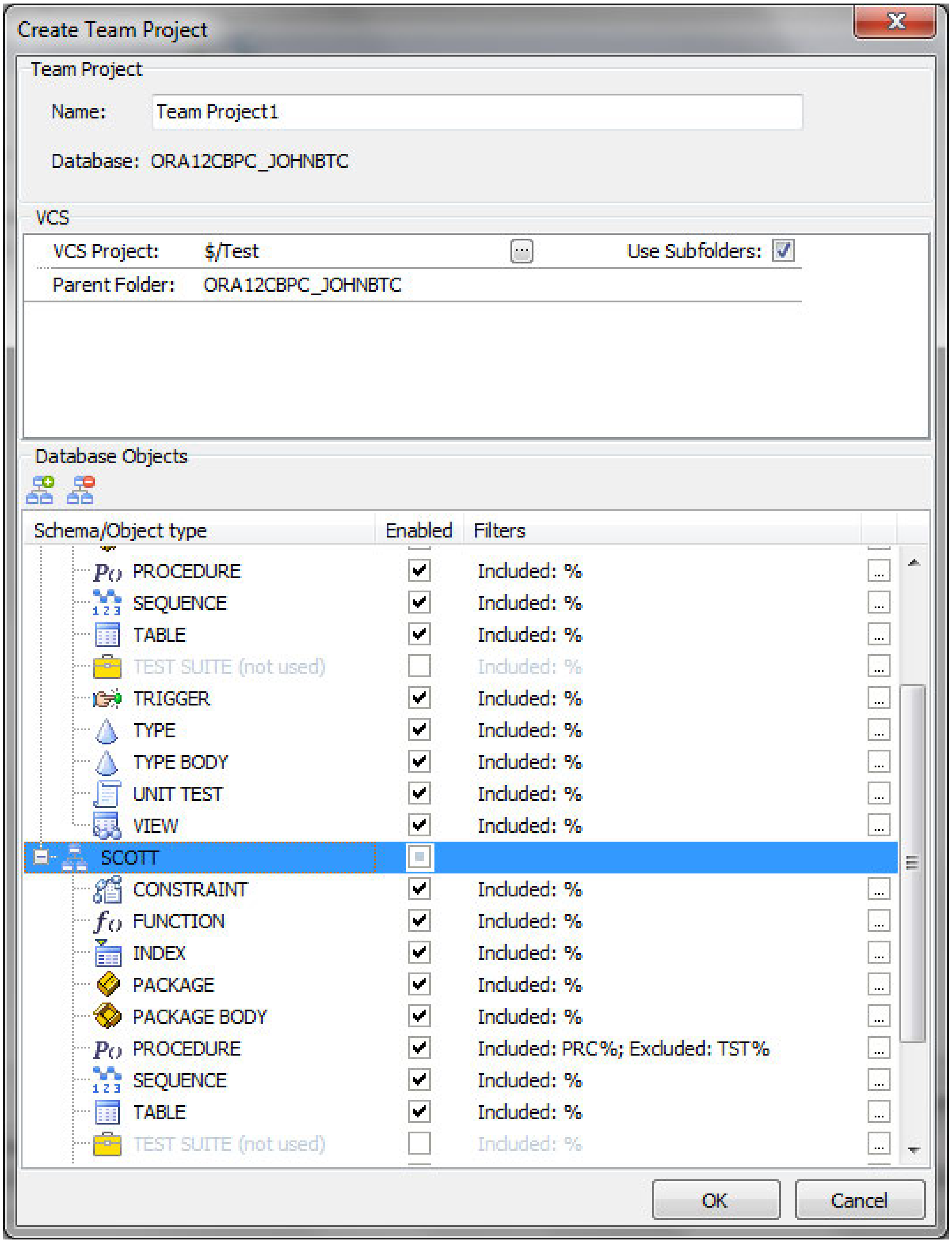
Here you can deselect any object types you do not wish to control. By default, all objects within those object types are included to be controlled by Team Coding.

Quest

You can also click the ellipses button next to the Filter column to specify both Include and Exclude filters on the object types in order to fine-tune which objects should be controlled by Team Coding. Clicking this button will display a window similar to the following:



Through this window, you can specify multiple filters for each column to fine-tune the control of your objects as needed. When you click OK, Team Coding will update your settings in the Team Projects page:



When you are satisfied with your Team Project settings, simply click the OK button.

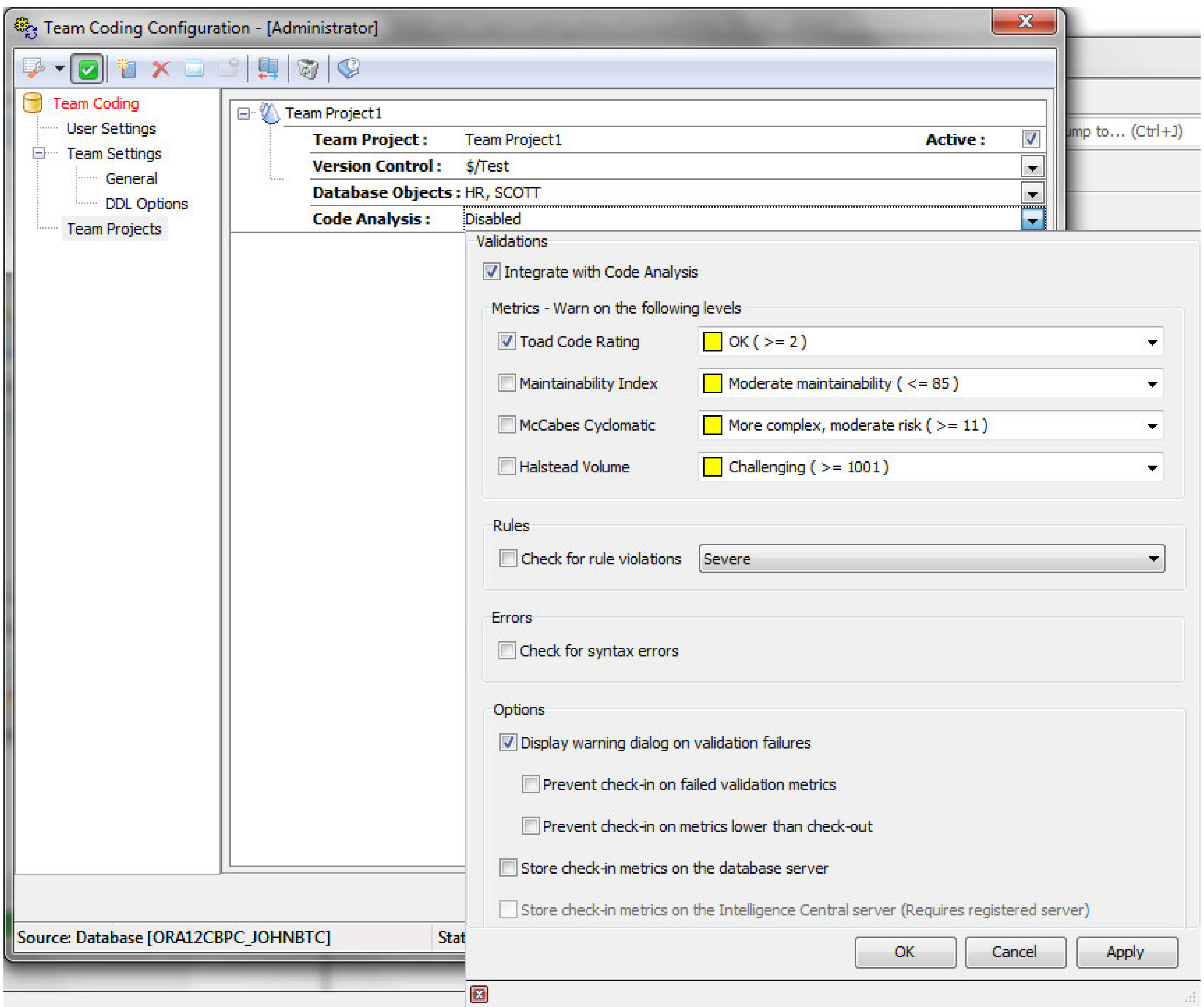
Note: You can specify multiple team projects if you desire; however, each Team Project must be linked to a separate project within your VCS provider. You cannot share a VCS project between multiple Team Projects within Toad.

Integrating with Code Analysis

Code Analysis (available with Toad Professional Edition and higher) allows you to help control and improve the quality of code being compiled into your database. Before an object can be compiled and/or checked in to Team Coding, a series of checks will be run against the code (code review) to help make sure the code being checked in follows common standard programming practices and is maintainable.

Toad uses standard code quality metrics, such as Halstead Volume, McCabes Cyclomatic, and Maintainability Indexes to determine code quality. Toad also combines these metrics into a Toad Code Rating, which is a composite of several rating criteria to provide a quick glance of the quality of the code. For more information on Code Metrics, see the ***About Code Analysis*** page in Toad's Help file.

To configure Team Coding to perform ***Code Analysis*** checks, click on the ***Code Analysis*** drop-down from the selected Team Project and click on the ***Integrate with Code Analysis*** checkbox.



Here, you can specify which metrics you wish to track, along with which level will cause the metric to fail and be reported back to the user before check-in. You can also specify a Code Analysis ruleset to check for potential code violations (for more information on rule sets and violations, see the Code Analysis section of Toad's help file).

You can also specify whether to make sure there are no syntax errors before check-in by selecting the ***Check for syntax errors*** option.

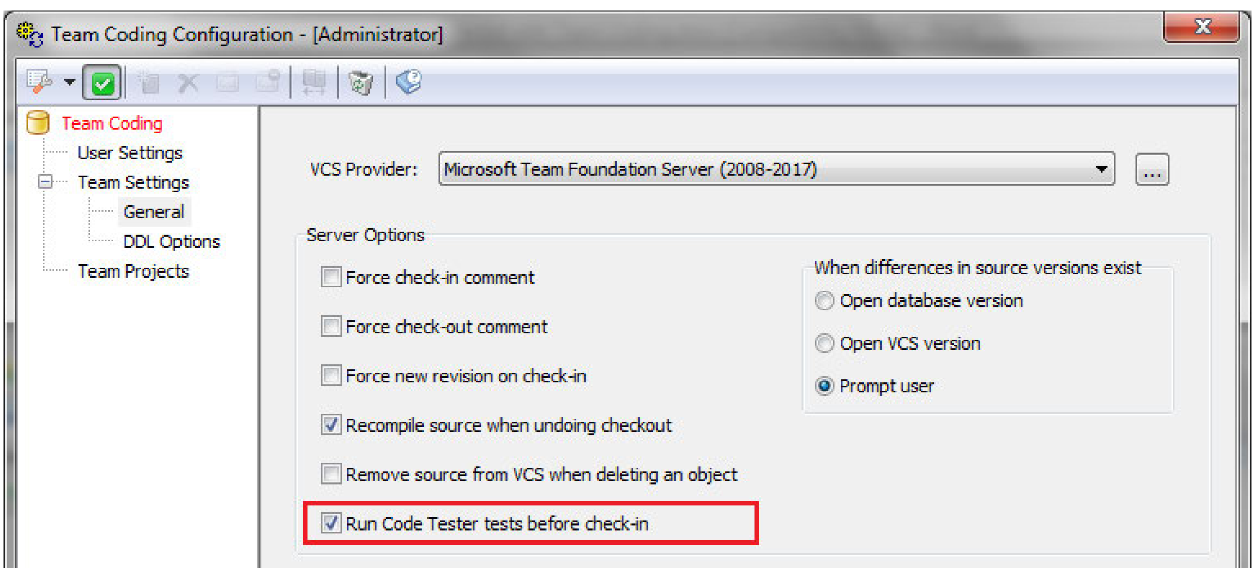
If the Code Analysis objects are installed on the server, or if you have Toad Intelligence Central installed in your environment, you can also choose to automatically upload the Code Analysis results to the server each time they're run for future reporting.

Integrating with Code Tester for Oracle

Code Tester for Oracle (available with Toad Development Suite) enables developers to automatically create, store and execute unit tests for their PL/SQL code simply by entering their known input parameters and specifying the expected outcomes. Unit tests are stored in the Code Tester repository, can be used for future regression testing and remain for the life of the code.

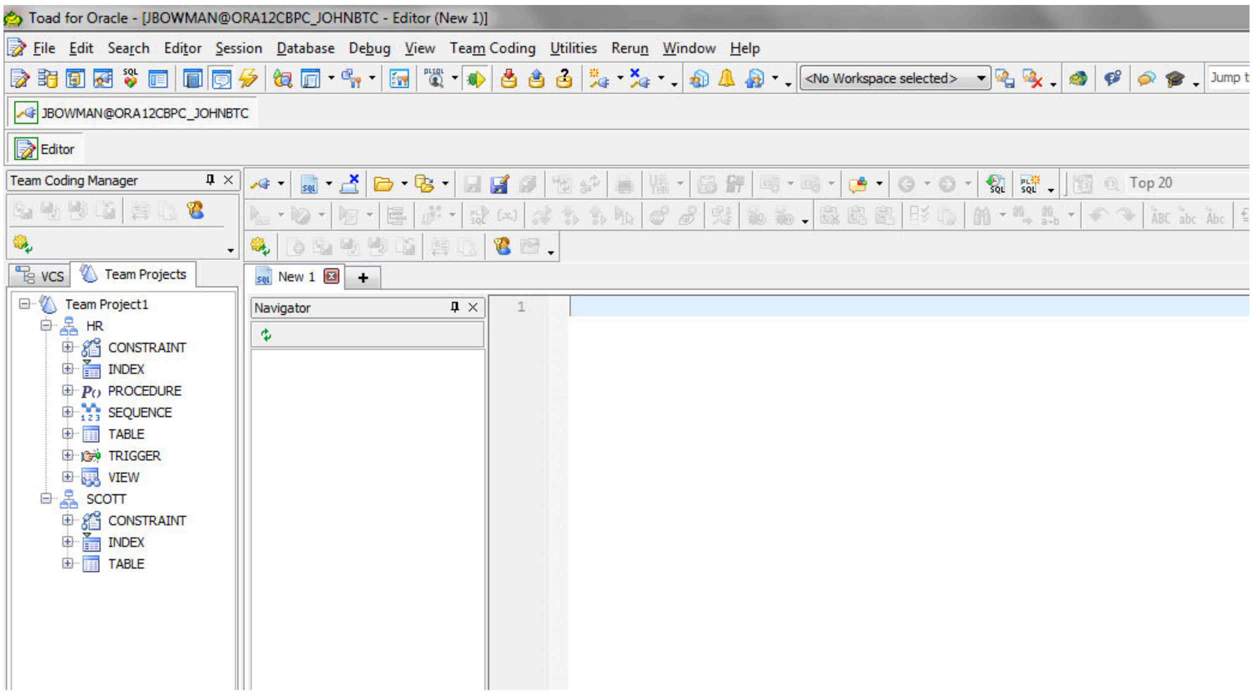
Since Toad for Oracle 2015 (v12.8) it has been possible to create unit tests in the Toad Editor making unit testing as easy as possible and a natural part of the developer's other tasks.

By integrating Code Tester with Team Coding, you can set a policy which enforces unit tests associated with the code being checked-in to be executed and a notification to be displayed if any tests or test cases fail (in which case, the code won't be checked in). This is designed to help prevent buggy code from reaching production.



Saving the Configuration

When you are satisfied with your Team Coding settings, simply click the OK button. Toad will save the Team Coding settings to the server and query the database for all objects under Team Coding's control. If a VCS provider has been defined, if will also prompt you to create a new base revision for any Team Projects that may have been created and/or modified. When finished, you will be able to see the list of controlled objects by selecting ***Team Coding Manager*** from the ***Team Coding*** menu. This will open the Team Coding Manager in the left-hand dock panel of Toad.

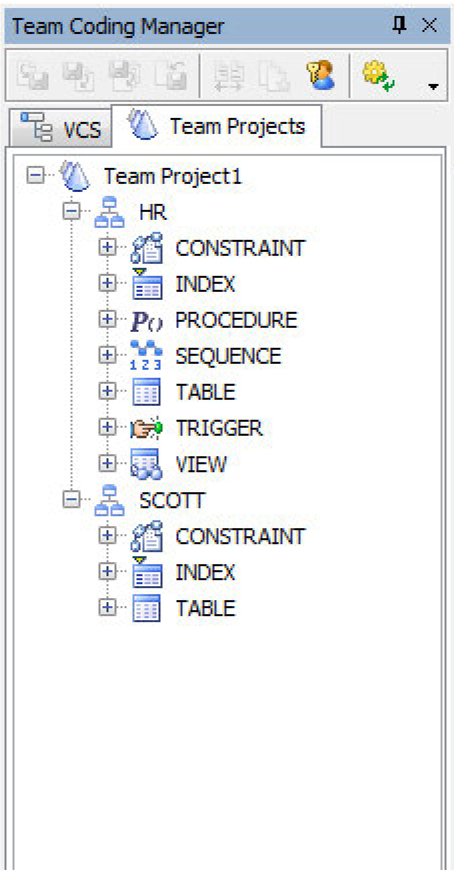


Note: When using the "All Schemas" option in your Team Project, the controlled objects will be built dynamically as you work with your database objects. As a result, you will not initially see any objects in your Team Projects tab of the Team Coding Manager. However, as you work with objects, checking them out and checking them in, they will automatically be added to Team Coding and will appear in the Team Projects tab of the Team Coding Manager.

Using Team Coding

Using the Team Coding Manager

The Team Coding Manager window provides a centralized place for working with database objects controlled by Team Coding. You can access this window by selecting ***Team Coding Manager*** from either the ***View*** or ***Team Coding*** main menus. This will bring up the following docked window on the left-hand side of Toad:



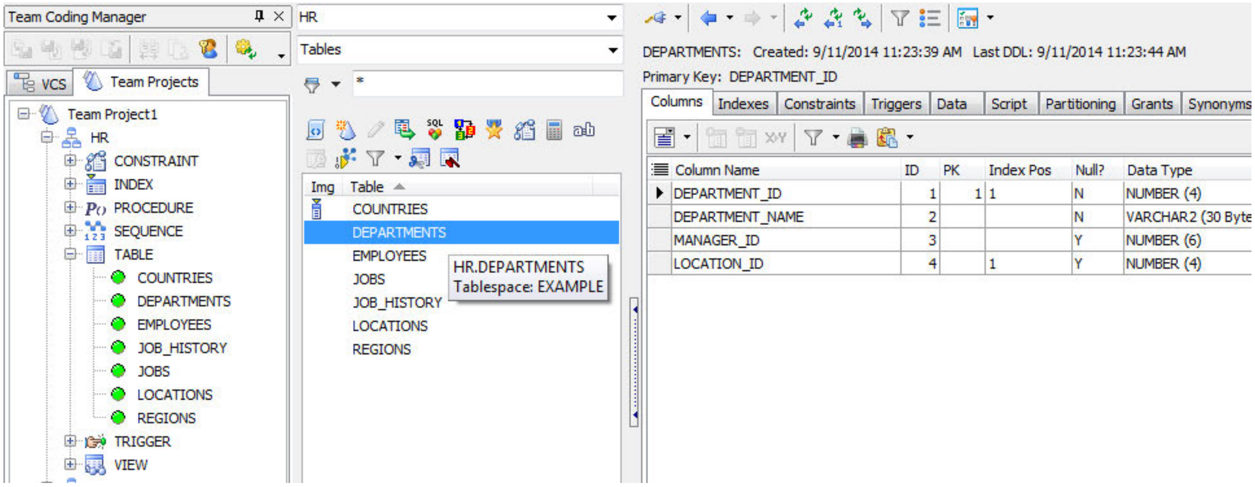
Note: If you linked a version control provider to your Team Project, you will see two tabs: ***VCS*** and ***Team Projects***. If you're working with Team Coding without a link to a version control provider, you will only see the ***Team Projects*** tab.

The ***VCS*** tab provides a way to browse through projects in your VCS provider. In this tab, you can open files, check files in and out of the VCS provider, compare the contents of the file between the working version and the VCS, and other VCS-related actions.

The ***Team Projects*** tab provides an organized way of working with objects controlled in the database. In this tab, you can open database object DDL in the editor, check objects in and out of the VCS provider, compare the contents of an object's source between the database and the VCS, create new revisions within the VCS, and other Team Coding object-related actions.

Working with Objects in the Schema Browser

Objects controlled by Team Coding will need to be checked out in the Schema Browser before they can be altered. By default, their edit buttons will be disabled until the object is checked out for editing, as shown in the following example:



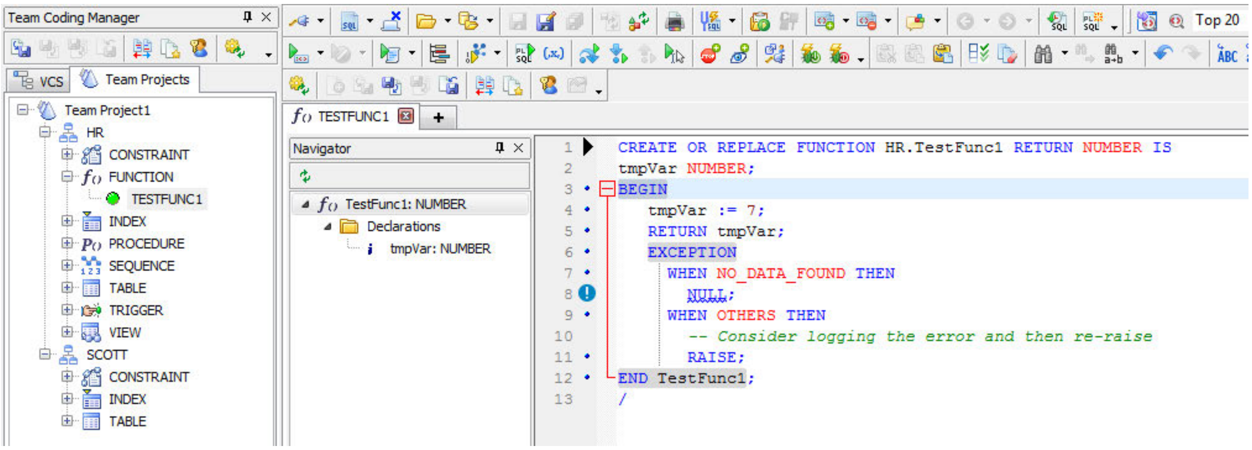
To check out an object, simply right-click on the object and select ***Team Coding –> Check-out.*** Alternatively, you can select ***Check-out*** from the ***Team Coding*** main menu. Once the object is checked out from Team Coding, the ***Alter*** button  will be enabled and will allow you to alter the database object.

When you are ready to check in your changes, simply right-click on the object and select ***Team Coding –> Check-in***. Alternatively, you can select ***Check-in*** from the ***Team Coding*** main menu.

If you decide against making your change, you can right-click on the object and select ***Team Coding –> Undo check-out***. Alternatively, you can select ***Undo check-out*** from the ***Team Coding*** main menu.

Working with Objects in the Editor

Similar to the Schema Browser, objects controlled by Team Coding will need to be checked out in the Editor before they can be compiled. The Editor has its own Team Coding toolbar which is visible by default, but can be disabled if you so choose:



Objects can be checked in and out from the Editor either by clicking on the respective tool button, by right-clicking in the Editor and selecting the appropriate action from the ***Team Coding*** context menu, or by selecting the appropriate action from the ***Team Coding*** main menu.

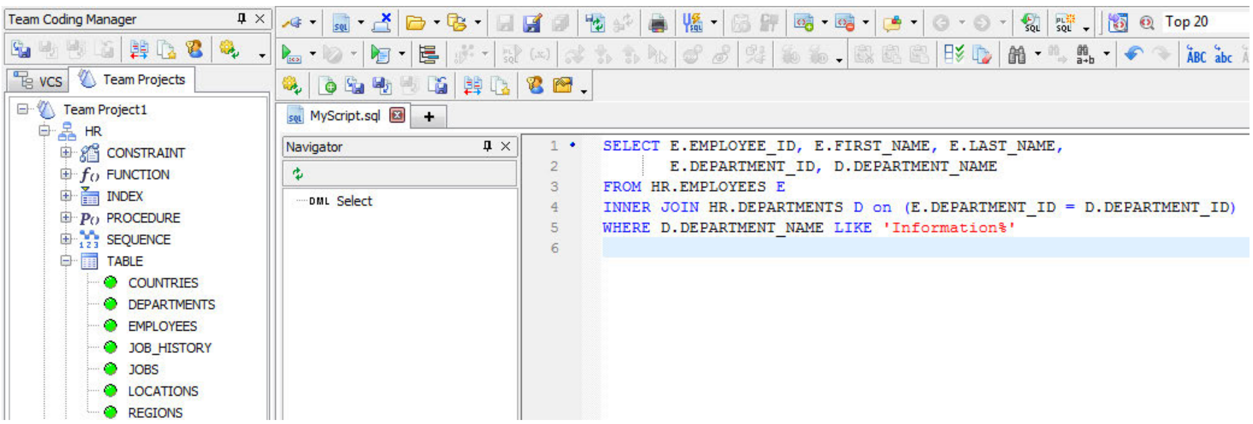
In addition, the Editor will automatically prompt you to check out a controlled object if it appears that the DDL for that object will be altered either from the ***Execute/Compile*** command or from the ***Execute as Script*** command.

Working with Scripts in the Editor

Team Coding also allows you to work with individual scripts and check those scripts in and out of the VCS provider. In this mode, the scripts are not controlled by the Team Projects that were set up previously, but are controlled manually by each user. This allows the user to organize their database scripts into whatever VCS folder structure they choose.

Note: Team Coding will continue to use the VCS provider specified within the Team Coding settings, but the user will be able to select their own project to store their scripts.

When the Editor senses that its contents represent a user script, it will automatically enable all VCS actions in the toolbar and menus, as in the following example:



Note: Files must be saved before these options are enabled.

In this mode, it is up to the user to handle all actions with the VCS provider manually. To do this, simply click the appropriate action in the Team Coding toolbar:

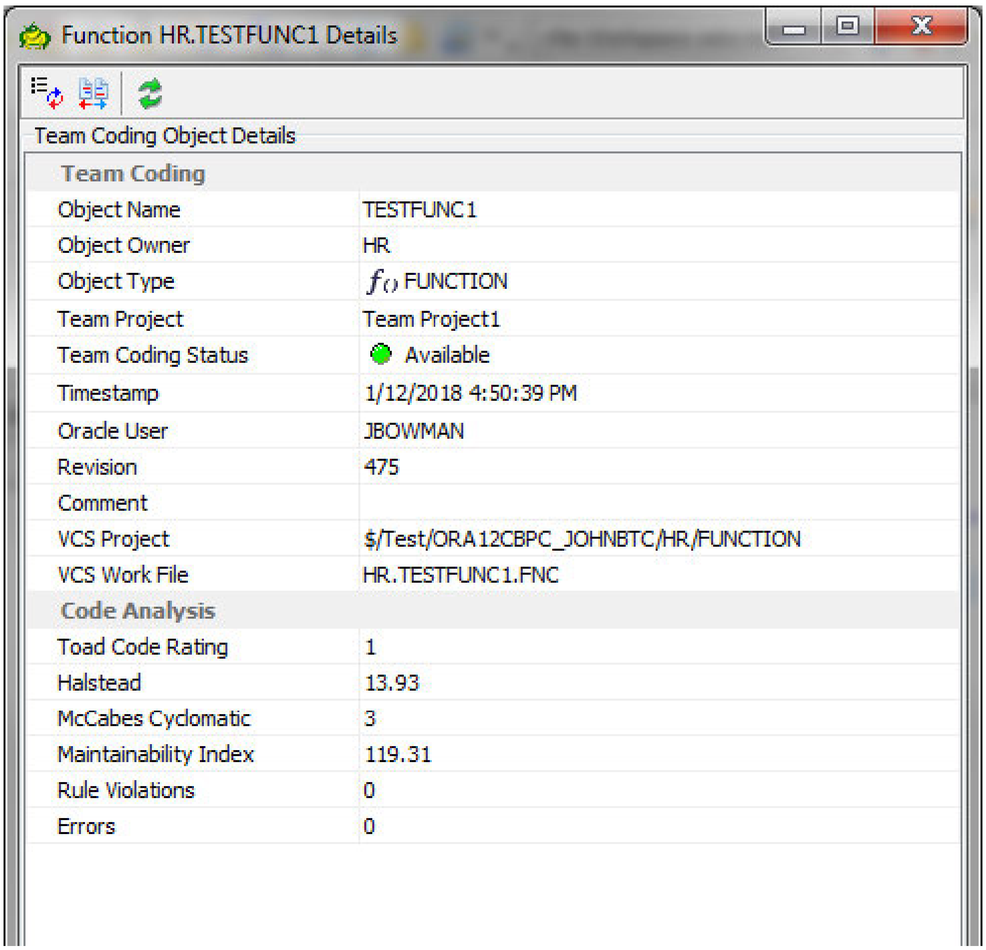
- Click the ***Select Project*** button  to select the VCS folder
- If the file doesn't already exist in the VCS, select the ***Add to VCS*** option
- If the file already exists in the VCS, select the ***Check-out*** option, make your changes, and then select the ***Check-in*** option.
- If the file is already checked out of the VCS, you can select either the ***Check-in*** or ***Undo check-out*** option, depending on your needs.

Note: When connected to a Team Coding database, Team Coding will attempt to automatically resolve local script folders to their corresponding VCS folder, if possible. If successful, the script will automatically be uploaded to its corresponding VCS folder. If the VCS folder cannot be automatically determined, the user will be prompted with the ***Select Project*** dialog before the ***Add to VCS*** or ***Check-in*** action is processed.

Viewing Team Coding Object Information

Viewing Information on a Single Object

Information about an object's status in Team Coding can be viewed in a number of places within Toad. At-a-glance information about an object's check-out status can be viewed directly within the ***Team Coding Manager***. To view detailed information about an object, right-click on the object in the ***Team Coding Manager*** and select ***Show Properties***. This will display the Object Details window, as in the following example:

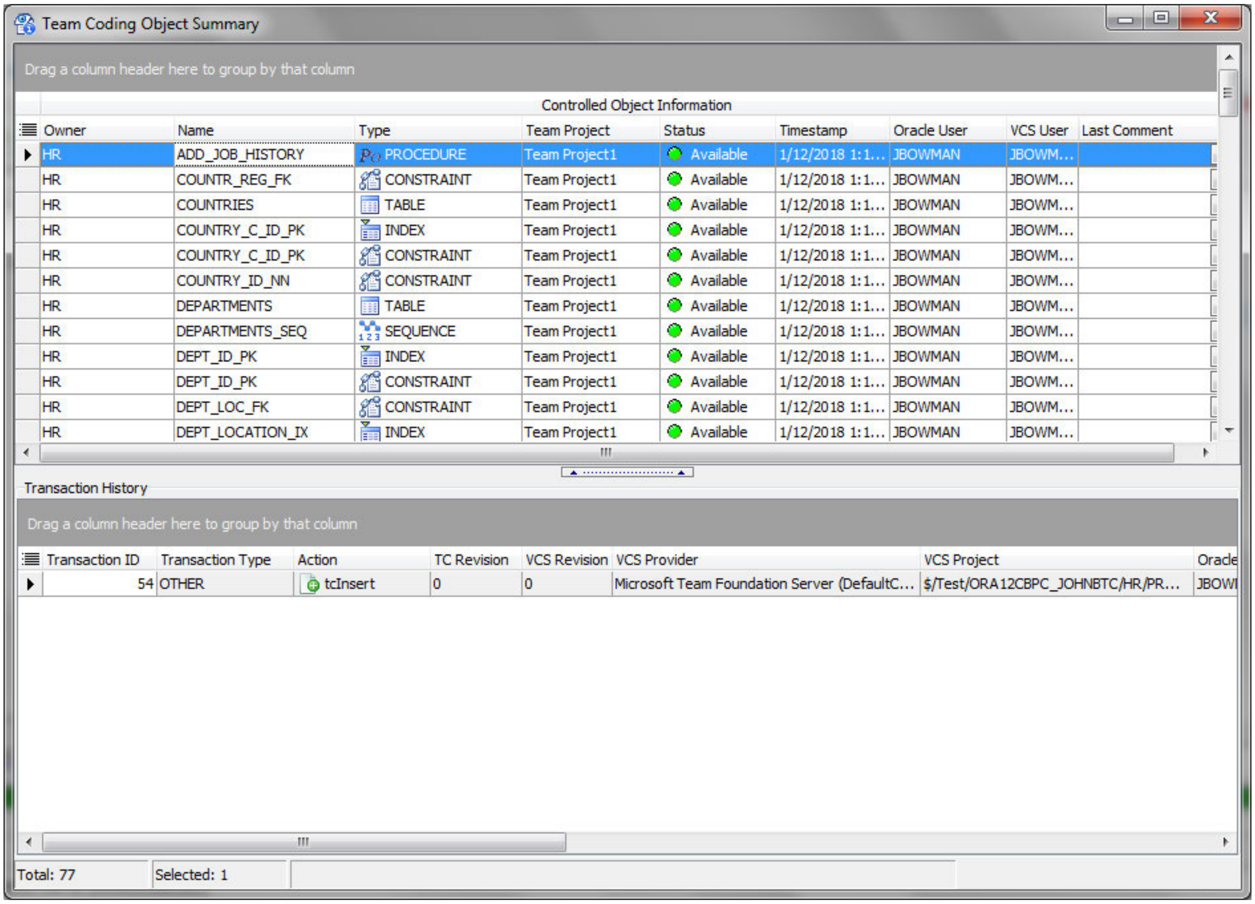


In this window, you can view details about the object as well as view revision history information within the VCS by clicking on the ***Display Revision History*** tool button  at the top of the form.

This window is also available from the Editor by enabling the ***Team Coding*** panel at the bottom of the Editor, and also from the Schema Browser, by selecting the ***Team Coding*** tab after selecting an object.

Viewing Information for All Objects

You can also view information about all objects controlled by Team Coding by selecting the ***Show Team Coding Objects*** option from the ***Team Coding*** main menu. This will open the ***Team Coding Object Summary*** window, as in the following example:



This window allows you to view the current status of all objects controlled by Team Coding, as well as view the transaction history for each object to show what actions were performed against the object, when those actions were performed, and which users performed them.