Oracle® Configuration Controls Governor

Integra Codebase User Guide

Release 4.2.2

September 2008

ORACLE

Oracle Configuration Controls Governor: Integra Codebase User Guide

Contents

1

Start Here

Integra Codebase provides change tracking, impact analysis, and versioning of both files and database objects, including Oracle E-Business Suite forms, reports, menus, libraries, and PL/SQL code. A component of the Oracle Configuration Controls Governor, Codebase provides superior tracking and reporting of changes – planned as well as executed.

Integra Codebase increases the effectiveness of development efforts while minimizing the challenges that cause project overruns and production downtime. Whether applying a patch, upgrading to a new release, or extending Oracle to meet specific business requirements, Codebase dramatically reduces the time and resources required to identify changes, dependencies, and impact on your environment.

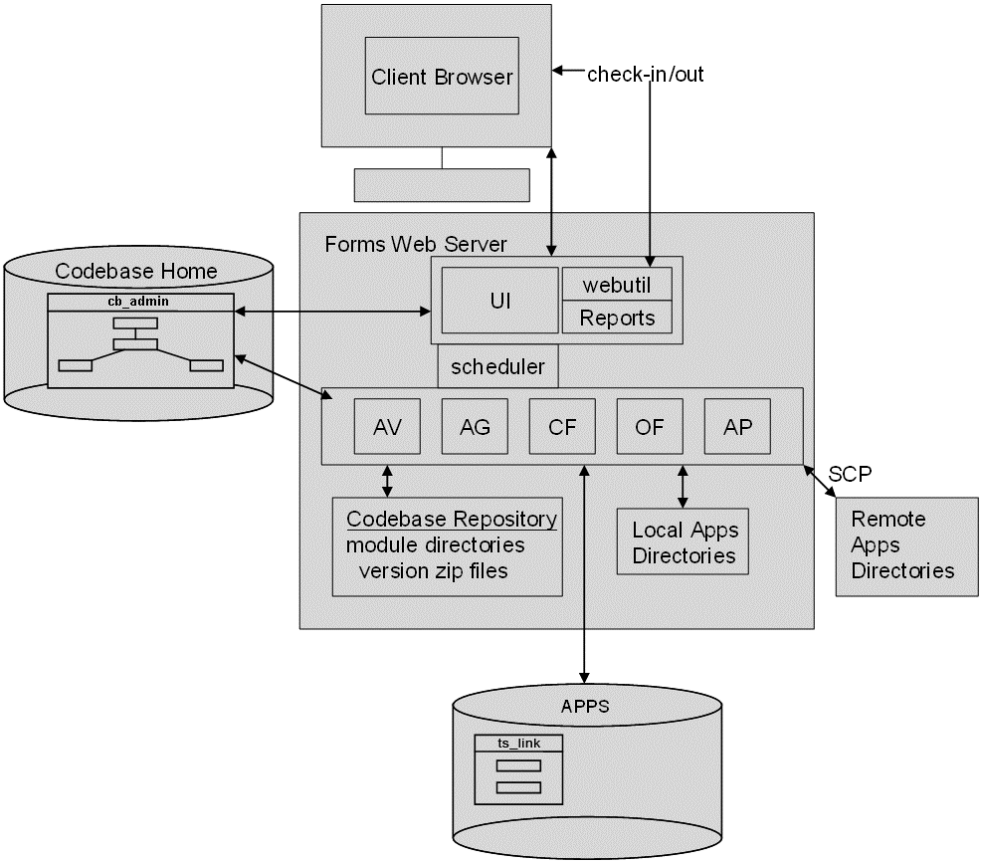
Integra Codebase's unique functionality provides the industry's deepest level of impact analysis. Using binary code comparison and dependency recognition, Codebase delivers robust documentation and analysis of your business applications' programmatic content.

Integra Codebase has eight major components:

- Maintain Versions (p. 36)**:** Interactive version reporting, comparison, and control, (including check-out/in, labeling, and restore), for all programmatic content types, including files, forms, reports, menus, libraries, and PL/SQL.
- Automated Versioning (p. 91)**:** Automatically versions files and database objects whenever they change. Populates Maintain Versions (p. 36).
- Automated Generation (p. 101)**:** Migration process that promotes files, creates executables, and installs database objects in an Environment.
- Change Finder (p. 65)**:** Interactive component that finds differences in files, database objects and database schemas.
- Automated Change Finder (p. 115)**:** Monitors files and schemas for changes.
- Publish (p. 90)**:** Creates documentation of programmatic content on demand.

- Automated Publishing (p 124)**:** Automatically creates documentation of programmatic content.
- Object Finder (p. 78)**:** Interactive component that lets you search for procedures, files, database object dependencies, subclassed forms, or attached libraries.

The following diagram depicts a typical installation managing an Oracle E-Business Suite environment:



Terminology

As you use Codebase, you'll encounter the following terms:

- **Codebase Home schema**: Database schema that stores all Codebase configuration data, and the information needed for comparing and versioning your files and database objects. Codebase can share this database with other applications.

  The schema resides in an Oracle database. The database, in turn, can reside on any server (including the Codebase Server). Codebase users and the Codebase Server access the schema using SQL*Net.

- **Codebase Server**: Runs the Codebase application, including its automated processes (Automated Versioning, Automated Publishing, Automated Change Finder and Automated Generation).
- **Environment**: Database instance monitored by Integra Codebase, and a set of files associated with that instance. Example: an Oracle E-Business Suite database instance, and the collection of programmatic files used by the Suite.
- **Module**: A single piece of programmatic content, such as a file or a database object (including Oracle E-Business Suite forms, reports, menus, libraries, and PL/SQL code).
- **Project**: Collection of Environments that are part of a code promotion hierarchy (e.g., Development, Test, and Production), and a set of user permissions and other configuration information. Projects are usually used to represent logical units of work as well as groups of Environments. The Project Administrator must create projects before anyone can use Codebase.

How to use Integra Codebase

To get started, create one or more Projects using the Maintain Projects window (p. 25).

Once you've done that, you have several options:

- For complete version control and change reporting, schedule Automated Versioning (p. 91). Once it has run, use the Maintain Versions (p. 36) to monitor and compare versions, and to check out/in content (aka "modules").
- If your organization uses another content versioning system, use Change Finder (p. 65) and/or Automated Change Finder (p. 115) to get detailed reports of changes.
- To promote modules from one Environment to another, schedule Automated Versioning (p. 91). Once it has run, set the modules' destination in the Maintain Versions window's (p. 36) **Promote to** field. Then schedule Automated Generation (p. 101) to perform the promotion. For more information, see Promotion (p. 54).
- To create reports that document your modules, use Publishing (p. 90) or Automated Publishing (p. 124).
- To search modules, use the Object Finder (p. 78).

If you've used Integra Codebase before

Integra Codebase offers many advantages over previous releases. It uses a new Java architecture that provides robust support for both Unix/Linux and Microsoft Windows Environments. It also has an improved user interface that's more reliable and easier to use.

The following chart summarizes the most significant differences between Codebase 3.9, 4.1, and 4.2:

	Codebase 4.2	Codebase 3.9 and 4.1
Operating System	Unix/Linux and Microsoft Windows	Microsoft Windows only
Usability	Streamlined and simplified user interface	
Performance	Standard J2EE processes replace Oracle Forms background processes, providing increased stability and reliability	
	Improved performance for all automated processes	
	Generation performance based on number of files generated	Performance related to number of files monitored
Environments and Objects	SSH and SCP allow management of unmounted directories; directory mounts are still supported	Managed file directories must be mounted on Codebase server
	Defining a promotion hierarchy is as simple as entering a list of Environments (p. 29)	A two-dimensional array of "Environments" and "Stages" must be configured

	Codebase 4.2	Codebase 3.9 and 4.1
	Assets can be promoted to any environment in the hierarchy	Assets must be promoted to each environment in the hierarchy, in the hierarchical order
	Same file name can appear in any number of directories	File name must be unique across all versioned directories
	Support for versioning and generating more database object types	
Generation	Generates only files marked for promotion	Unexpectedly generates files not marked for promotion (to match destination environment to internal image, which can be out of sync with actual source)
	Improved logic to determine whether generation succeeded for User Defined Extensions (UDEs)	
	Expanded set of UDE substitution variables	
	Support for more object types	
	Command logs available within generation logs	Command logs available on server only
Creating documentation	New Publishing component (p. 90) provides documentation on demand	Documentation must be generated using Automated Publishing

	Codebase 4.2	Codebase 3.9 and 4.1
Reports	Reports are available on the workbenches that generate them, ready for decision support and troubleshooting	To view reports, you must leave your current workbench and navigate to other windows
	New reports: *Environment* (p. 106) - Module versions promoted to an environment *Unpromoted Content* (p. 106) - Modules never promoted to an environment *Comparison* (p. 106) of modules promoted to two different environments *Audit* (p. 106) of files promoted to an environment vs. actual file system contents *Change Finder Summary* (p. 65) *Job Summary* (p. 140)	
Jobs	More flexible scheduling options	
	Schedule any series of automated processes in a single step using the Schedule window (p. 142)	
	No browser required on Codebase server	Must run an Oracle Form in a browser window on Codebase Server

	Codebase 4.2	Codebase 3.9 and 4.1
	A single summary email message is sent describing all changes made during a job	Separate email messages sent for each file acted upon
	The View Inside Jobs window's Log window (p. 140) replaces the Action Summary window, consolidating job status and results	Job status and results are displayed in two different windows
Workstation	Web browser	*3.9 only:* Oracle Forms client
Application server	10gAS Release 2	*3.9* - 6i Forms client/server *4.1* - 10gAS Release 1

As mentioned above, reports are now available on relevant workbenches:

Codebase 4.1	Codebase 4.2
View Reports > View Automated Change Finder Reports >	
View Customizations	Maintain Versions: Compare To... (p. 50)
View Version Changes	Maintain Versions: Compare To... (p. 50); Change Finder (p. 65)
View Oracle Changes	Change Finder: Compare Time Periods (p. 76)
View Error Summary	View Inside Jobs: View Log (p. 140)
View Schema Changes	Change Finder: Compare Specific Modules (p. 77)
View Reports > View Automated Publishing Reports >	
View Document	Automated Publishing (p. 124)
View Error Summary	View Inside Jobs: View Log (p. 140)

Installation Status >

View File Status Print Environment	Environment report (p. 106)
Reprocess File	Maintain Versions (p. 36): **Promote to** field (promote again to the same Environment)
View Errors	View Inside Job: View Log (p. 140)

2

Login

All Codebase users must begin by logging in using the **Welcome to Integra Codebase** window.

To display that window, open a web browser window and visit:

10gAS Release 1 users:

`http://`*`hostname`*`:`*`port`*`/forms90/f90servlet?config=`*`codebase`*

10gAS Release 2 users:

`http://`*`hostname`*`:`*`port`*`/forms/frmservlet?config=`*`codebase`*

..where:

`hostname` is the domain name of the web server configured to serve Codebase

`port` is the port used to access Codebase on *`hostname`*

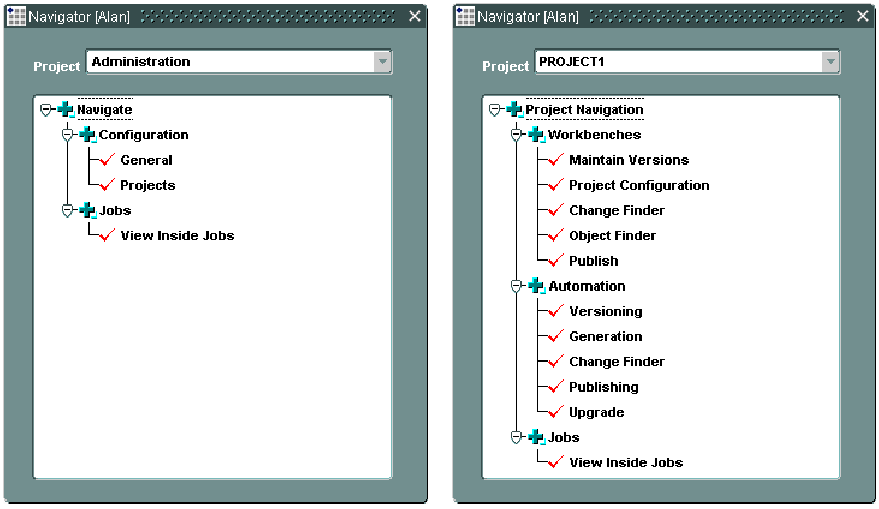
`codebase` is the **Application Configuration Name** chosen during installation (usually `codebase`). For more information, see "Integra Installer Worksheet" in the *Integra Codebase Installation Guide.*

The window has these fields and buttons:

Username	Your Codebase Administrator creates your username.
Password	Your Codebase Administrator creates your password. To change it, log in, then select **Action > Change Password** from the menu bar.
Login (button)	Verifies your username and password, and displays the Navigator (p. 10).
Cancel (button)	Cancels login and closes this window.

3

Navigator



The Navigator is your gateway to all Codebase windows. To open a window, double-click its name.

The windows listed depend on the setting of **Project**, and on your roles and permissions:

Window	Required Role(s)
Configuration:	
General (p. 14)	`Administration`
Projects (p. 25)	`Administration` *AND* `Projects`

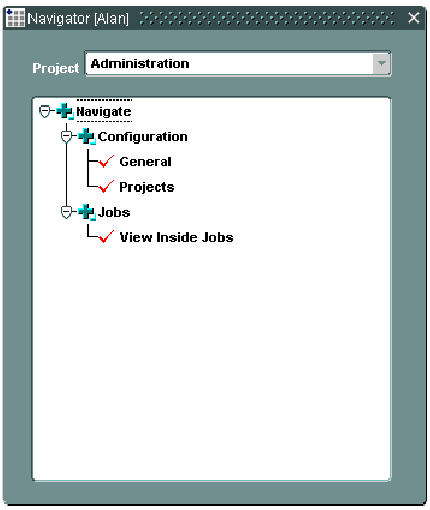
Workbenches:	
Maintain Versions (p. 36) Change Finder (p. 65) Object Finder (p. 78) Publishing (p. 90)	`Projects`
Project Configuration (p. 57)	`Projects` *AND* Projects: Users: `Admin` (p. 27)
Automation:	
Versioning (p. 91) Generation (p. 101) Change Finder (p. 115) Publishing (p. 124)	Projects *AND* Projects: Users: `Admin` (p. 27)
Jobs:	
View Inside Jobs (p. 136)	Projects: Users: `Jobs` (p. 27)

The window contains one field:

Project	You must select a project prior to double-clicking the name a Codebase window. The dropdown contains only the projects that you have been assigned to (for information about project assignment, see Projects: Users (p. 27)). If you have been assigned the **Administrator** role, the **Administration** project is listed (for information about role assignment, see General: Users (p. 17)).

4

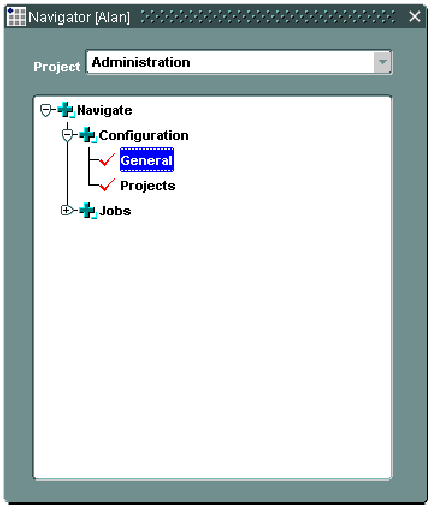
Administration Role



When **Project** is set to **Administration**, the Navigator window (p. 10) lists the windows that let you configure Integra Codebase, its users, and its projects.

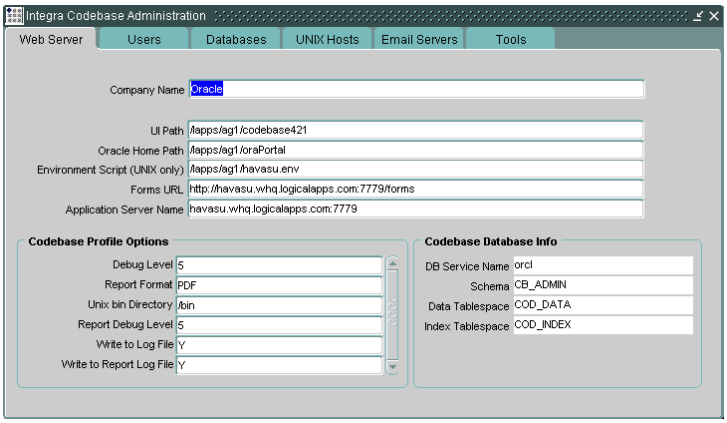
In This Section

General



The **General** window lets you configure Integra Codebase's global settings.

Web Server



This tab contains three groups of fields:

Company Name	Name of your company or organization.
UI Path	Full path on Codebase Server to directory where Integra Codebase is installed.
Oracle Home Path	Full path to the Oracle Home of the application server that serves Codebase.

Codebase Profile Options

Debug Level	Level of detail in Codebase's application server log. Enter an integer between `1` and `5`; `1` is least and `5` is most. We recommend using `1` or `2`. Has no effect when **Write to Log File** is set to `N`.

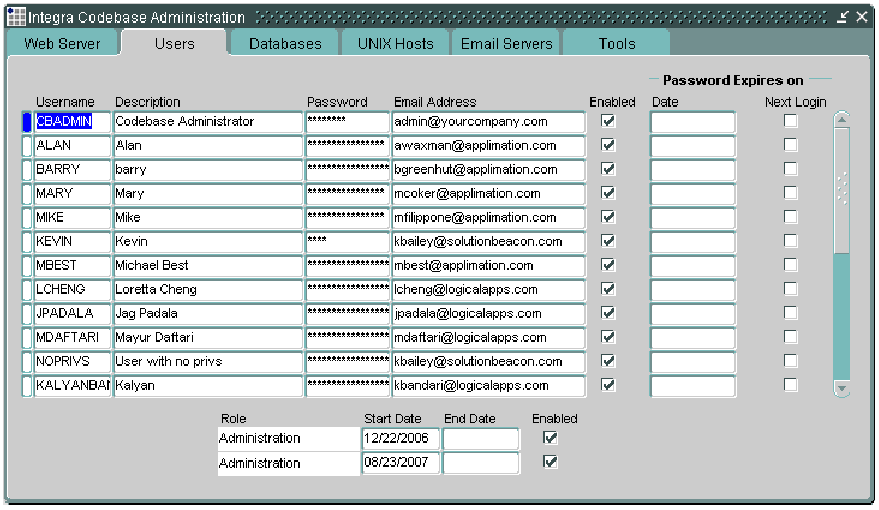
Report Format	Default format for reports. The options are:
	`Delimited`
	`HTML`
	`PDF`
	`RTF`
	`XML`

Unix bin Directory	***Unix/Linux users:*** Full path on Codebase Server to directory where the Unix/Linux utility `sh` is stored. ***Windows users:*** Leave blank.
Report Debug Level	Level of detail in Codebase's report error log. Enter an integer between `1` and `5`; `1` is least and `5` is most. We recommend using `1` or `2`. Has no effect when **Write to Log File** is set to `N`.
Write to Log File	Enter `Y` to direct the application server to log Codebase messages; `N` to turn off logging.
Environment File	***Unix/Linux users:*** Full path on Codebase Server to directory where the application server user's environment file is stored. For more information, see "Before You Install: System Requirements: Codebase Server: Configuration" in the *Integra Codebase Installation Guide.* ***Windows users:*** Leave blank.

Codebase Database Info

DB Service Name	Service name of the database that houses the Codebase schema.
Schema	Name of the Codebase schema.
Data Tablespace	Tablespace that houses the Codebase schema's tables.
Index Tablespace	Tablespace that houses the Codebase schema's indexes.

Users



This tab lets you add and manage Codebase user accounts.

To create a user:

1 Click any **Username** or any empty row, and click  (**Row > Insert**). A blank row is highlighted.

2 Fill in the blank row's fields, and click  (**Action > Save**).

3 Set the user's **Roles** as desired, and click  again.

To change a user's information:

4 Change the settings of the user's fields, and click  .

To disable a user:

Once a user has been created, it cannot be deleted, but it can be disabled, which prevents login:

1 Uncheck **Enabled** to the right of the user's **Username**, and click  .

The tab contains the following fields:

Username	String used by Codebase to identify the user, and entered by the user when logging in. The string may not contain spaces or any of these characters: `! | / = $ & @ " ' ` ,`
Description	Typically the user's full name.
Password	The password entered by the user when logging in.
Email Address	The user's email address.
Enabled	When checked, the user can log in.

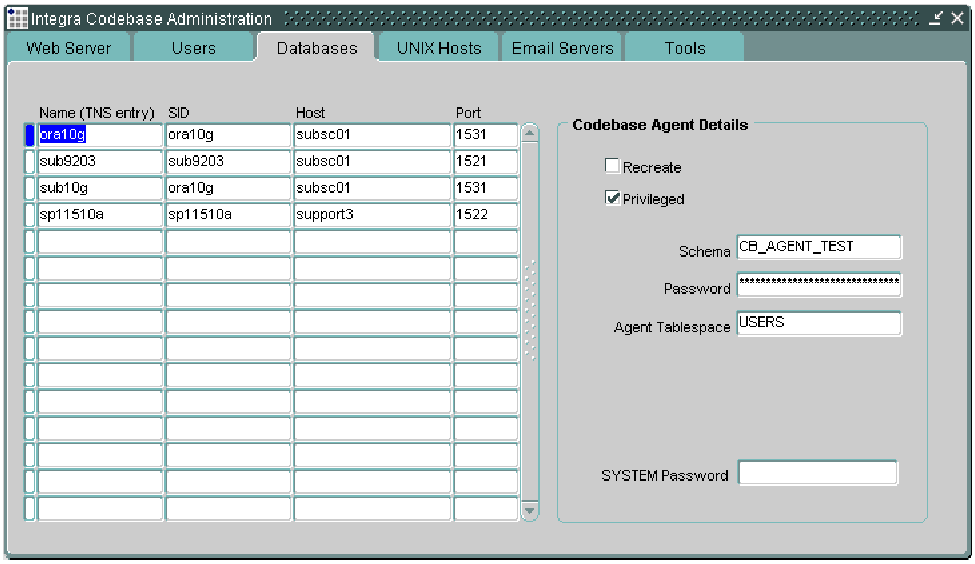
Password Expires On

Date	The user will be asked to choose a new password the first time s/he logs in after this date.
Next Login	The user will be asked to choose a new password the next time s/he logs in.

Roles

Role	The user's roles. Roles determine the windows a user can see. There are two roles: `Administrator` and `Project`; for more information, see Navigator (p. 10).
Start Date	The user will have the Role any time s/he logs in after this date.
End Date	The user will no longer have the Role any time s/he logs in after this date.
Enabled	When checked, the user has the role (unless prevented by **Start Date** or **End Date**).

Databases



This tab lets you identify the databases that Codebase can manage.

To add a database:

1 Click any **Name** row, and then click (**Row > Insert**). A blank row is highlighted.

2 Fill in the blank row's fields, and click (**Action > Save**).

3 Optional: Specify tablespaces in the **Tablespaces** area, and click again.

To change a database's settings:

1 Change the settings as desired, and click .

To permanently remove a database:

1 Remove all references to the database on Automated Versioning's Schemas tab (p. 95), and Automated Generation's Schemas (p. 107) and File Locations (p. 108) tabs.

2 Click the database's **Name**, and then click (**Row > Delete**).

The tab contains the following fields:

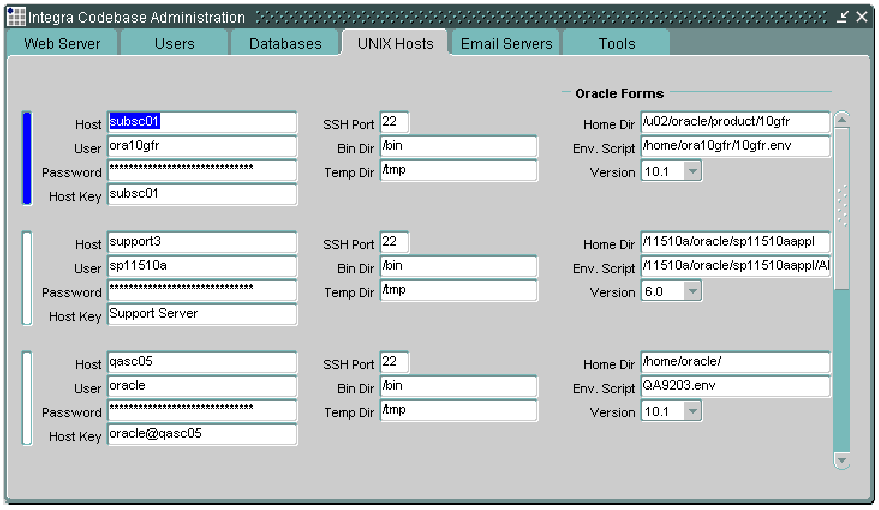
Name (TNS entry)	The database's TNS entry on the Codebase Server.

SID	The database's SID.
Host	The domain name of the database host.
Port	The port for accessing the database via the **Host**.

Codebase Agent Details

Schema	Codebase creates an empty schema in the database, with this name.
Password	The password for accessing **Schema**.
Agent Tablespace	Tablespace where Agent will temporarily store data. At least 10MB of free space must be available. The tablespace can be shared with other applications, or dedicated to Codebase.
Recreate	When checked, the Codebase Agent is updated. Use this when: An Environment housed in the database has been refreshed. You are updating a password.
Privileged	When checked, the Codebase Agent can auto-version schemas that are not normally accessible to **Schema**.
SYSTEM Password (to create Codebase Agent)	Password of the selected database's SYSTEM user, which is used to create the Codebase Agent.

Unix Hosts



Use this tab to specify all Unix/Linux hosts used to connect to Environments' file system directories and databases (the latter are specified on the **Databases** tab).

> ***Microsoft Windows users:*** Windows Environments do not require **Hosts**. If all databases managed by Integra Codebase are hosted in Windows Environments, leave this tab blank. If some are hosted in Windows and some in Unix/Linux, enter hosts only for databases hosted in Unix/Linux.

To add a host:

1 Click on any **Host** row, and then click (**Row > Insert**). A blank row is highlighted.

2 Fill in the blank row's fields, and click (**Action > Save**).

To change a host's settings:

1 Change the settings as desired, and click .

To permanently remove a host:

1 Remove the databases (p. 19) served by the host.

2 Click the **Host**, and then click (**Row > Delete**).

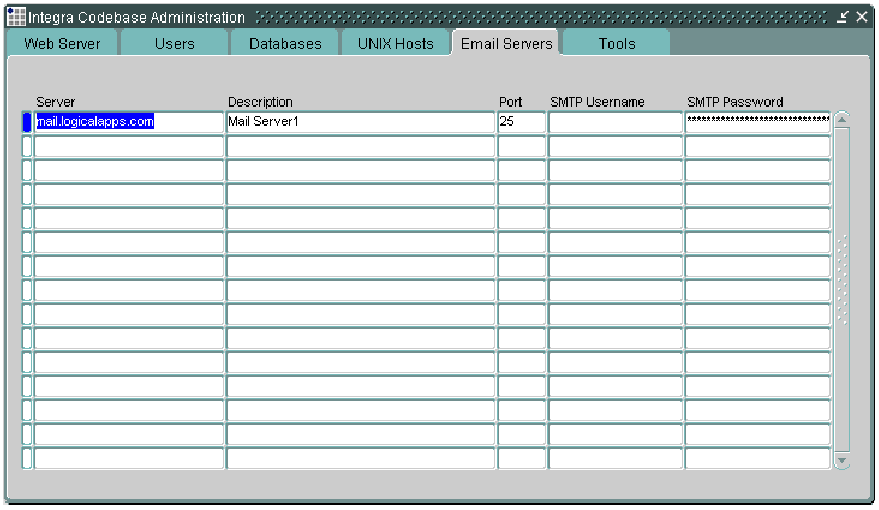
The tab contains these fields:

Host	The host's domain name.
Port	Port for accessing the database Codebase will manage.
User	Codebase logs in as this user when transferring files, using `ssh` and `scp`. Therefore, this user will own the files.
Password	**User**'s password.
Bin Dir	Full path on host to the directory that holds the operating system's typical BIN utilities.
Temp Dir	Full path on host to directory where Codebase can store files temporarily.

Oracle Forms

Home Dir	***Oracle Forms applications (including E-Business Suite) users:*** Full path on host to Oracle Forms Home directory. ***All others:*** Leave blank.
Version	***Oracle Forms applications (including E-Business Suite) users:*** Version of Oracle Forms installed on host. ***All others:*** Leave blank.
Env. Script	***Oracle Forms applications (including E-Business Suite) users:*** Full path on host to Oracle Forms compilation environment file. ***All others:*** Leave blank.

Email Servers



This tab lets you configure the email servers used by Integra Codebase and the Codebase Agents to send email messages during automated operations (for more information, see Projects: Email setup (p. 32)).

To add a server:

1 Click any **Server** row, and then click (**Row > Insert**). An empty row is highlighted.

2 Fill in the empty row's fields, and click (**Action > Save**).

To change a server's settings:

1 Change the settings as desired, and click .

To permanently remove a server:

1 Remove all references to the server on Projects' Email setup tab (p. 32).

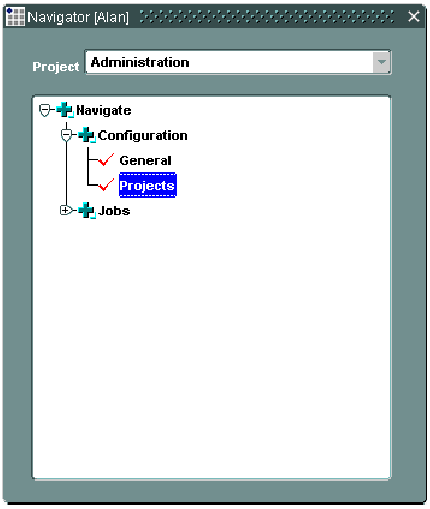
2 Click the Server, and then click (**Row > Delete**).

The tab contains these fields:

Server	Your outgoing (SMTP) email server. Most SMTP servers have the following naming convention: `smtp.xyz.com`

Description	Brief description that identifies the server.
Port	Port used to send email via your outgoing SMTP server.
SMTP Username	If your SMTP server requires you to login, enter a valid username. Some SMTP servers do not require login information when you are already connected to their network.
SMTP Password	If **SMTP Username** is required, enter the password.

Projects



The **Projects** window lets Codebase Administrators set up projects and assign users to them. You must have both the `Administrator` and `Project` privileges to access this window.

Projects let you organize logical units of work. At its core, each project is a collection of Environments that are part of a code promotion hierarchy (e.g., Development, Test, and Production), and a set of user permissions. For example, each department should have its own Integra Codebase project, in order to provide control and security.

A Codebase Administrator must create at least one project before anyone can use Codebase's features.

If you have questions about the processing of files or database schemas, and are not a Codebase Administrator, contact an Administrator for information.

The upper portion of the window contains fields for choosing projects.

Project	The project's name. This name will appear in the title bar of each window when a project user is logged in.
Description	Could be used to describe why the project was created, or what content or activities are monitored.

To create a project:

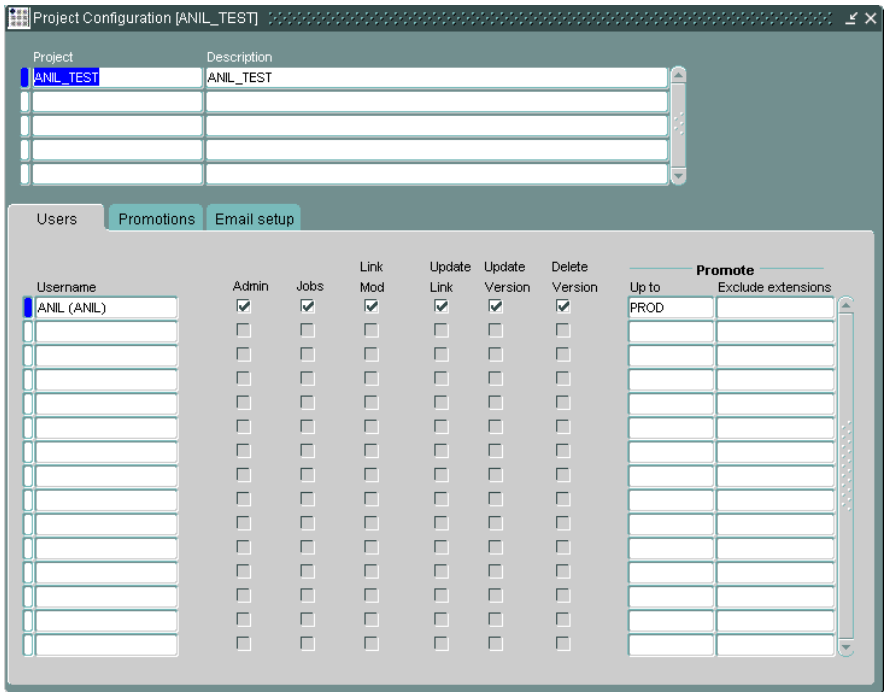
1 Click any **Project** row, and then click  (**Row > Insert**). A blank row is highlighted.

2 Fill in the blank row's fields, and click  (**Action > Save**).

To edit a project:

1 Click once on its name; the project's details appear in the tabs below.

Users



This tab lets you give Codebase users permission to use the project, and control what they can do with the project.

To add a user to a project:

1 Select the **Project** in the upper portion of the window.

2 Click on any **Username**, and then click (**Row > Insert**). A blank row is highlighted.

3 In the blank row, select the **Username**, fill in the other fields, and click (**Action > Save**).

To change a user's privileges:

1 Select the **Project** in the upper portion of the window.

2 Revise the user's fields, and click .

To remove a user from a project:

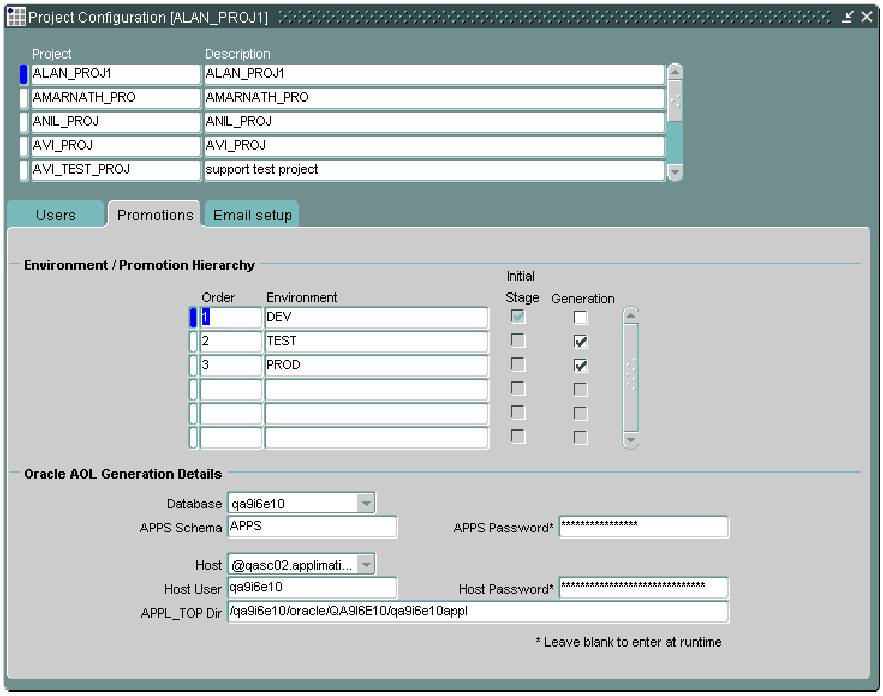
1 Select the **Project** in the upper portion of the window.

2 Click on the Username, and then click (**Row > Delete**).

The tab contains the following fields and buttons:

Username	Select the user who will have permission to use the project.
Admin	Grants access to the Project Configuration (p. 57) window.
Jobs	Grants the ability to run the automated processes (by enabling the **Schedule** button on the **General** tabs of their windows). Consider creating a user with only the `Projects` role, and giving this user the sole ability to schedule all projects' automated processes. By limiting scheduling capability to a single user, you simplify auditability, and decrease the chance of scheduling conflicting or duplicative processes (potential conflicts are described in each process' documentation). Users are created on the Administration: Users tab (p. 17).
Link Mod	Grants the ability to create records in the Link Modules window (p. 51).
Update Link	Grants the ability to update records in the Link Modules window.
Update Version	Grants the ability to update records (e.g., versions) in the Maintain Versions window.
Delete Version	Grants the ability to delete versions (p. 49) in the Maintain Versions window.
Promote: Up to	Limits how far the user can promote modules (p. 54).
Promote: Exclude extensions	Comma-delimited list of file extensions that the user will NOT be able to promote.

Promotions



This tab lets you define the promotion hierarchy, which is followed when promoting code from your development Environment to other Environments (and eventually to your production Environment). Here is an example of a simple promotion hierarchy:

1. Development
2. Test
3. Production

> If your project will use Automated Versioning, but not the promotion mechanism, do not enter any records in this tab.

The **Initial Stage** checkbox indicates where all code will originate; only one of these checkboxes can be checked. The Initial Stage Environment should be the one where developers are creating code.

When defining the promotion hierarchy, the Environments must abide by the following rule: each non-initial-stage Environment (e.g., TEST, PROD) must correspond to an Initial Stage Environment (e.g., DEV) by containing matching Compilation Schemas (see Automated Generation: File Locations (p. 108)).

To define a project's promotion hierarchy and/or AOL details:

1 Select the **Project** in the upper portion of the window.

2 Set the tab's fields as desired, and click  (**Action > Save**).

The tab contains these fields and buttons:

Define Environment/Promotion Hierarchy

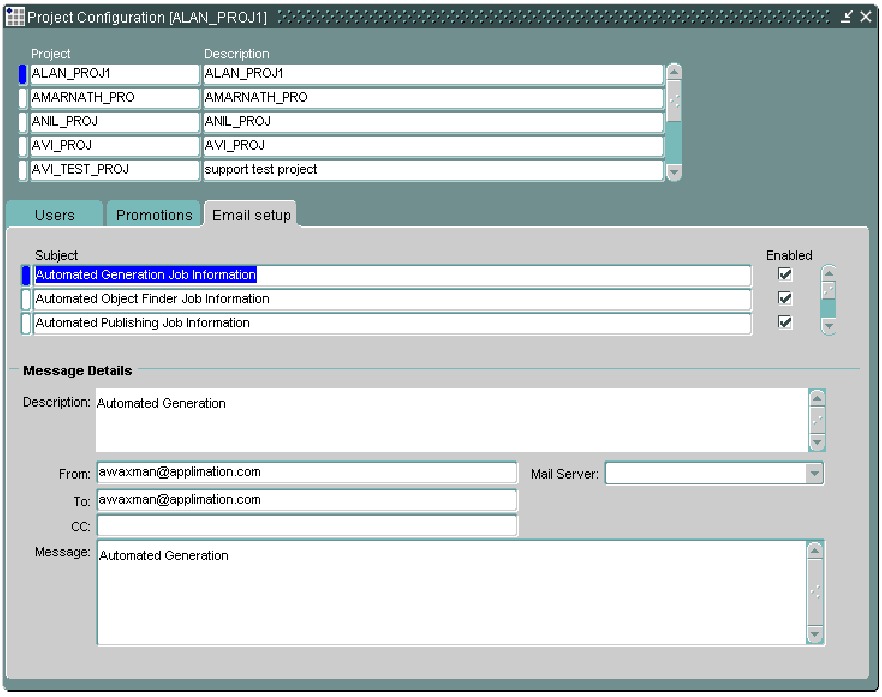
Order	Order in which code is promoted, starting with the Initial Stage (e.g., Development).
Environment	Name of the Environment.
Initial Stage	When checked, identifies the Environment that code originates from (i.e., the one where development occurs). See above for more information.
Generation	When checked, Automated Generation (p. 101) can perform all of its functions on the Environment. When unchecked, Automated Generation only promotes files; it does not run SQL or compile forms.

Oracle AOL Generation Details

Database	TNS entry on the Codebase Server for the Environment's database.
APPS Schema	Name of the Oracle E-Business Suite APPS schema (typically `APPS`).
APPS Password	Password of the APPS schema. Optional; if you do not enter a value, it will be requested at runtime.
Host	***Unix/Linux only:*** Host used to access the **Database**.

Host User	***Unix/Linux only:*** Username for logging into **Host** (typically `APPLMANAGER`). Enabled if **Host** is chosen.
Host Password	***Unix/Linux only:*** Password for logging in as **User**. Enabled if **Host** is chosen. Optional; if you do not enter a value, it will be requested at runtime.
APPL_TOP Dir	***Unix/Linux only:*** Full path on host to the `APPL_TOP` directory. Enabled if **Host** is chosen.

Email setup



This tab lets you configure the email messages generated by the project.

To create a message:

1 Select the **Project** in the upper portion of the window.

2 Click any **Subject** row, and then click (**Row > Insert**). A blank row is highlighted.

3 Fill in the blank row's fields, and click (**Action > Save**).

To edit a message:

1 Select the **Project** in the upper portion of the window.

2 Click the message's **Subject**.

3 Revise the message's fields, and click .

The tab contains these fields:

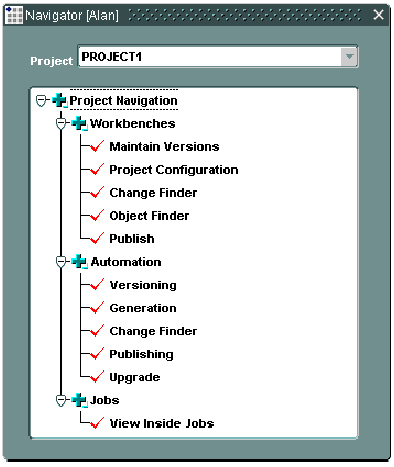
Subject	The message's "Subject" header.
Enabled	When checked, the message can be sent; when unchecked, the message cannot be sent.

Message Details

Description	Short description of the message. Appears only on this tab; does not appear in the message.
From	Usually the email address of the Project Administrator, so developers' replies will be sent back to the Administrator.
To	Email address(es) used when no address can be obtained for the developer associated with a file in the Link Modules (p. 51) or Maintain Versions (p. 36) windows. Separate multiple email addresses with commas.
Cc	Email address(es) that should get a copy of the message.
Mail Server	Outbound email server to use. Email servers are specified on the Administration window's Email Servers tab (p. 23).
Message	The message's body. All **Subject** variables (listed above) can be used in **Message**, as well as: `sub_comments` – **Comments** in Maintain Versions (p. 36) for the file being processed. `sub_obj_names` – List of objects containing PL/SQL code that are no longer inherited. This runtime variable is available only when Automated Publishing (p. 124) sends a warning message that PL/SQL code is no longer inherited.

5

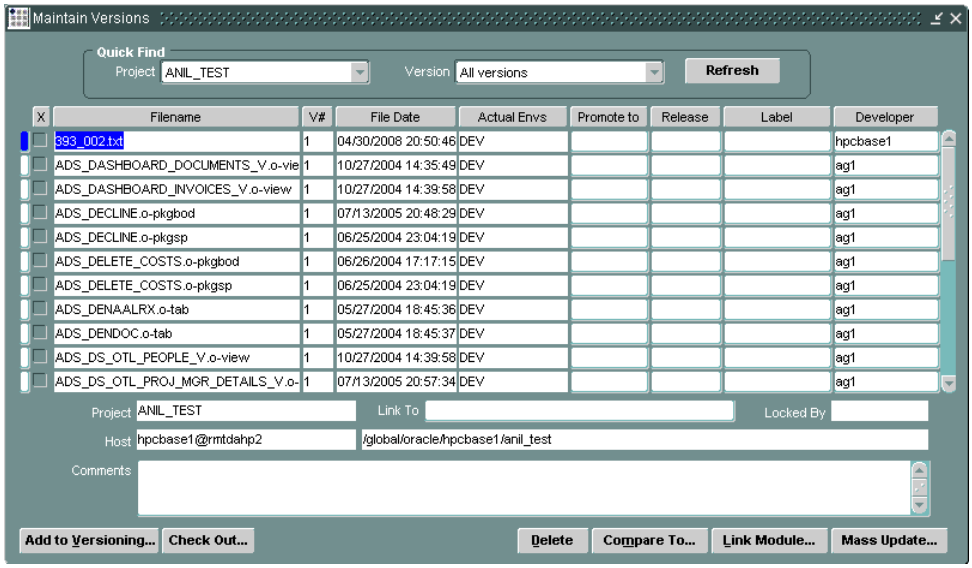
Projects Role



When **Project** is set to any project name, the Navigator (p. 10) lists the windows that let you use the project, and configure project automation. The **Workbenches** let you perform Codebase's manual day-to-day operations. **Configurations** let you set up and manage Codebase's project automation.

In This Section

Maintain Versions



This window lets you work with the file and database object versions created by Automated Versioning (p. 91). You can use it to:

- Add a file to versioning (p. 43)
- Find a version (p. 41)
- View a version (double-click its **Filename**)
- Compare two versions (p. 50)
- Add a release number, label and/or comments to a version (p. 41)
- Update a field for many versions (p. 53)
- Link a file to a version (p. 42)
- Associate a custom file with a standard file (Link Modules (p. 51))
- Delete a version (p. 49)
- Check versions out (p. 45) to prevent others from making simultaneous changes, and check the changed versions back in (p. 47)
- Schedule the promotion of versions to other Environments (using the **Promote to** field, described below; Automated Generation (p. 101) performs the actual promotion during its next run)

Your abilities depend on the privileges granted to you by your Codebase Administrator (in Projects: Users (p. 27)):

All users assigned to a project are allowed to view the window's information and the versions themselves. They can also view the log (p. 140), which contains error messages when files cannot be versioned (e.g., because they are either opened or locked).

Users with the **Update Version** privilege can update the following fields: **Rev Num**, **Label Name**, **Developer**, and **Comments**.

Users with the **Delete Version** privilege can delete versions.

Users can promote versions up to Environment specified in their privileges. Users cannot promote versions that have filename extensions specifically excluded by their privileges.

The window contains these fields and buttons:

Quick Find: Project	Filters the modules (i.e., files and database objects) listed below. This list contains all projects you have been assigned to, and one other choice: **All projects**.
Quick Find: Version	Filters the modules listed below. This list contains all version numbers created by Automated Versioning, and two other choices: **All versions** and **Latest version**.
Quick Find: Refresh (button)	Refreshes the modules displayed. Click this button whenever you suspect that modules have changed since the last time you selected a **Project** or **Version** on the left.
X	Shows whether the module has been checked out (and therefore locked).
File Name	Name of the module. This is a query-only field. Double-clicking this field displays the module, using the application associated with it in your web browser.

V#	Version number created by this window or Automated Versioning.
File Date	Modification date at the time the module was versioned. This is a query-only field.
Actual Envs	Environments in the promotion hierarchy where the version can be found. Populated after Automated Generation runs. In order for this field to show accurate information for the Initial Stage Environment (e.g., DEV), Automated Versioning must be run prior to Automated Generation (i.e., DEV may not appear on the latest version of a file if the version was created after Automated Generation processed the file).
Promote to	To schedule the promotion of this version, select the destination Environment from the dropdown. This dropdown lists only the Environments that you are allowed to promote to. You may promote the same version to an Environment more than once. Files will not be promoted to a non-initial stage Environment unless the corresponding Initial Stage Environment has already processed the file. Therefore, when adding a brand new file and immediately promoting it to another Environment, you must run Automated Generation for the Initial Stage Environment before running it for the non-initial stage Environment.
Release	Can be used to indicate a major release number. The field will default to the value entered in the **Major Release Number** field on Automated Versioning's General tab (p. 93). You should set the default value before starting new development on a new major release.

Label Can be used to identify modules that must be promoted as a group. After a developer assigns a Label to each version that needs to be promoted, s/he would typically pass the Label to the person who will promote the modules. That person can quickly identify the modules to be promoted by querying this field.

To view a list of all values entered in this field, click the field and press `Ctrl-L`. Use the list of values to retrieve an existing Label, or to see whether a proposed Label name already exists.

Developer Name of the developer responsible for maintaining this version.

A value can be entered manually, but this field is also automatically updated:

When the module is checked out or in using this page's **Check Out/In** button, **Developer** is set to the user who checked it out or in.

When the module is versioned by Automated Versioning (p. 91), and the module is a file, **Developer** is set to the file's operating system owner.

When the module is versioned by Automated Versioning, and the module is a database asset, Codebase creates a file corresponding to each version, and sets **Developer** to the operating system user that it uses when creating the files.

To view a list of all users in the current project, and all other Developer values entered in this field and in the Link Modules window, press `Ctrl-L`.

If **Developer** is a Codebase username with Type=`Project User`, Codebase can retrieve and use the user's email address (which is assigned in the Projects window (p. 25)).

Project The module's project. This is a query-only field.

Host	The host and directory where the file was versioned. This is a query-only field.
Link To	Lets you link the version to a document, tracker request, or any other external file. You can enter a full path and filename directly in the field, or select **Edit > Edit Field…** from the menu bar for a larger editing area. To open a linked file, double-click the **Link To** field. The file will open based on the operating system file extension association.
Locked By	The user who checked out the version.
Comments	Comments about the version. Anyone given access to this window can update this field.
Add to Versioning... (button)	Displays the Add a File to Versioning window (p. 43).
Check Out/In (button)	Click on the desired version, and then click this button to open the Check Out (p. 45) or Check In (p. 47) window.
Undo Checkout (button)	*Enabled only if you have selected a version that you have already checked out.* Cancels the checkout.

Delete (button)	Delete the version from the window, and upon clicking the **Save** button, the corresponding record in `tsversions` will be updated by setting `deleting_for=USER`. This action marks the record to be deleted by the Automated Versioning process. The Automated Versioning process will delete all records where `tsversions.deleting_for IS NOT NULL`. For each record that is deleted by Automated Versioning, the corresponding version is also deleted out of the archive file. This method is used to accomplish the following objectives: Developers having access to delete versions via the Maintain Versions screen no longer need write access to the archive directory. This provides better security for the archives since only the Automated Versioning process needs write access. Performance is greatly increased when a user deletes ten versions or more.
Compare To... (button)	Displays the Compare To window (p. 50).
Mass Update (button)	Displays the Mass Update window (p. 53).

Find a Version

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Select **Query > Enter Query** from the menu bar.

3 Enter values to match in any column(s), in the first row where version information normally appears.

4 Select **Query > Run Query** from the menu bar. The **Maintain Versions** window displays matching records.

To once again view all records, enter and run a query without any values.

Add Release, Label and/or Comments

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 To enter a version's release number: Click the **Release** field and enter a value.

3 To enter a label: Click the **Label** field and enter a value.

> Press `Ctrl-L` to see a list of all values entered in this column.

4 To enter a comment: click the **Comments** field and enter your comment.

5 Click  (**Action > Save**).

Link a File to a Version

You can link a file to any version. For example, you can link a program specification document to a version.

> This functionality works on Microsoft Windows workstations only.

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Click the **Filename** of the desired version.

3 Click the **Link To** field. Select **Edit…Edit Field** to open the **Link To** window.

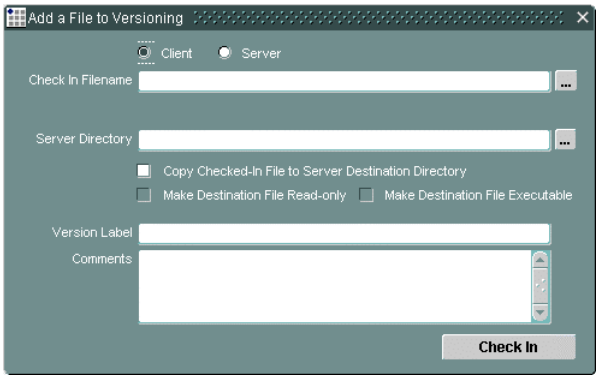
4 Click the **Browse** button and select the file that you want to link.

5 Click **OK** to close the **Link To** window.

6 Click  (**Action > Save**).

> To open a linked file, double-click the **Link To** field.

Add to Versioning



To add a file to versioning:

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Click **Add to Versioning**. Integra Codebase opens the **Add to Versioning** window.

3 Enter the following information:

Client Server	Current location of the file. `Client` is your workstation.
Check In Filename	File to add to versioning.
Server Directory	Full path to directory where file will be checked in.
Copy Checked-In File to Server Directory	When checked, copies the file to **Server Directory**. If you do not check this box, and do not copy the file manually to **Server Directory**, Automated Versioning (p. 91) will not version the file.
Make Destination File Read-only	When checked, Automated Versioning will write-protect the file so it cannot be updated.

Make Destination File Executable	***Unix/Linux users:*** Sets the owner's Executable permission to TRUE. If the file system is on a remote server, the owner is the Unix/Linux user that Codebase logs in as when working with the file system, which is set on the Administration window's Hosts tab (p. 21)). If the server is accessed via NFS, the owner is the NFS user.
Label	Sets the version's **Label** field.
Comments	Description of file.
Check In (button)	Applies your settings. To discard your settings without adding the file, close this window.

4 Click **Check In**.

To cancel the addition process, close the window.

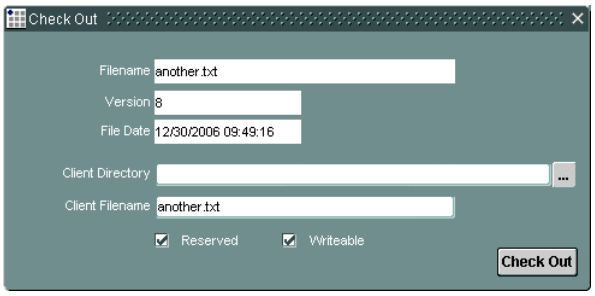
Check Out

Integra Codebase lets you check out a version, revise it, and check it back in (with an incremented version number). You can cancel a checkout at any time. Generally, a version is locked while checked out (see **Reserved** below).

To check out a version:

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Click the **Filename** of the version that you want to check out, and then click **Check Out**. Integra Codebase opens the **Check Out** window.



3 Enter the following information:

Filename **Version** **File Date**	*These display-only fields show the current file version from the main window.*
Client Directory	Enter the appropriate output location where the file should be put. This field defaults to the last location where the content was versioned by Automated Versioning.
Client Filename	Enter the filename to use for the checked-out file. This field defaults to the filename used when the file was last versioned by Automated Versioning.
Reserved	Check this box to lock the version.
Writeable	Check this box if you want the file to be writeable when put into the file system. If you do not check this option, the file will be read-only when put into the file system.

Check Out (button)	Applies your settings, puts the version into the file system, closes this window, and refreshes the Maintain Versions window (p. 36). To abandon checkout, close this window before you click this button.

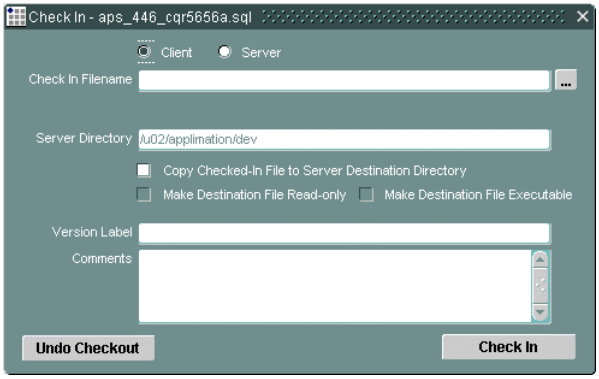
4 Click **Check Out**. Integra Codebase responds with a message indicating the success of your action.

To cancel the checkout process, close the **Check Out** window before you click its **Check Out** button**.**

To undo a checkout later, select the version in the Maintain Versions window (p. 36), and click **Undo Checkout**.

To check the version back in after you make changes, see Check In (p. 47).

Check In



To check in a version that you checked out earlier:

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Click the **Filename** that you want to check in, and click **Check In**. Integra Codebase opens the **Check In** window, listing the **Version, File Date, File Name** and **Extension** of the **File Name** that you are checking in.

3 Enter the following information:

Client Server	Current location of the version. `Client` is your workstation.
Check In Filename	File to check in.
Server Directory	Full path to directory where file will be checked in.
Copy Checked-In File to Server Directory	When checked, copies your modified version to the directory that Automated Versioning is monitoring. If you do not check this box, and do not copy your modified version to its original **File Location** (shown in the Maintain Versions window (p. 36)), Automated Versioning will not version the file.
Make Destination File Read-only	When checked, Automated Versioning will write-protect the file so it cannot be updated.

Make Destination File Executable	***Unix/Linux users:*** Sets the owner's Executable permission to TRUE. If the file system is on a remote server, the owner is the Unix/Linux user that Codebase logs in as when working with the file system, which is set on the Administration window's Hosts tab (p. 21)). If the server is accessed via NFS, the owner is the NFS user.
Label	Sets the version's **Label** field.
Comments	Description of the changes made.
Undo Checkout (button)	Discards your changes and unlocks the version.
Check In (button)	Applies your settings. To discard your settings without checking the version in or undoing the checkout, close this window.

4 Click **Check In**.

To cancel the check-in process, close the window.

Delete

The **Maintain Versions** window lets you delete a version if your username has the **Delete Version** privilege. Integra Codebase's performance increases if a given module (i.e., piece of versioned content) has less than ten versions.

Follow these steps to delete a version:

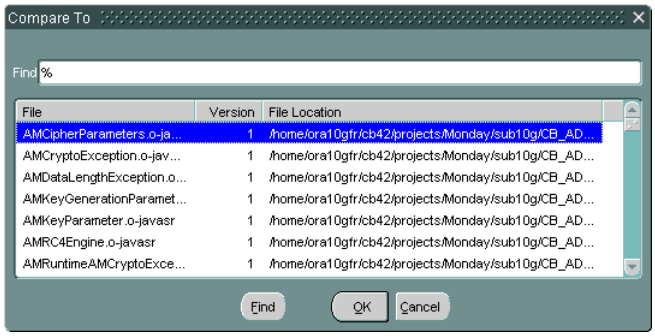
1. Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.
2. Click the **Filename** that you want to delete, and then click **Delete Version**. Integra Codebase responds with a message verifying if you want to continue.
3. Click **Yes**.
4. Click  (**Action > Save**) to commit your action and mark the version to be deleted by Automated Versioning during its next run.

Compare To

To compare two versions:

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Click the **Filename** that you want to view changes for, and click **Compare To…**. Integra Codebase opens a list of values displaying the versions that can be compared.



3 Choose the version you want to compare and click **OK**. The report appears.

Link Modules

This window lets you associate custom files with standard files (e.g., Oracle E-Business Suite files) and/or developers.

Automated Change Finder and Automated Generation use these links to determine which files to process, and how to process them (if your Project uses the **Process Only Linked Modules** option).

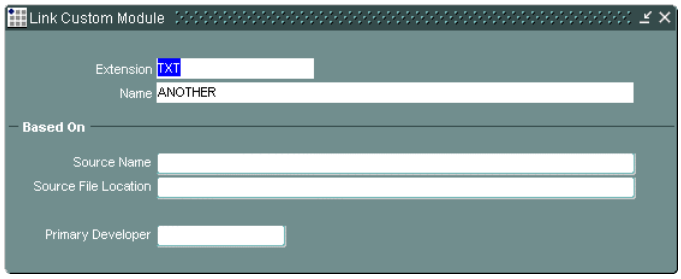
> The links apply to all of the project's Environments.

To create or modify a link

> All users can view the links, but only users with the Projects: Users tab's (p. 27) **Link Mods** privilege can add links, and only users with that tab's **Update Links** privilege can modify an existing link.

1 Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.

2 Click **Link Modules**. The **Link Modules** window appears.



3 Enter this information:

Extension	Filename extension of custom module (i.e., file or database object) to be linked. Options include `SQL` and `TXT`.
Name	Filename (excluding extension) of custom module to be linked.

Source Name	Filename (excluding extension) of the standard module on which you based your custom module. Enter `TEMPLATE` if you created the module from scratch.
Source File Location	Full path to the directory containing the source file.
Primary Developer	Developer responsible for maintaining the custom module. Press `Ctrl-L` to choose from a list of the project's developers.

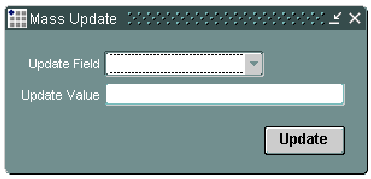
4 Click  (**Action > Save**).

Mass Update

This window lets you update a field's value for all versions currently found in the Maintain Versions window (p. 36) (even undisplayed records).

To perform a mass update:

1. Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.
2. Perform a query to display the versions that you want to update. If the query returns some versions that you do not want to update, you can remove them from list by selecting them, and then selecting **Row > Clear** from the menu bar.
3. Click the **Mass Update** button. The **Mass Update** window appears.



4. Set **Update Field** to the field you want to update, and **Update Value** to the value you want to assign. When the cursor is in the **Update Value** field, you can usually press `Ctrl-L` to view a list of existing values (although some fields do not have such a list).
5. Click **Update**.
6. Once Integra Codebase updates the fields, click  (**Action > Save**).

> To cancel the update, perform a new query without saving.

Promotion

Prior to using Automated Generation (p. 101) to promote files, generate executables and/or install SQL in your database, Automated Versioning must version the source content (aka "modules"). Once versioned, you must mark the modules for promotion in the Maintain Versions window's (p. 36) **Promote to** field.

Automated Generation automatically places the specified versions into the appropriate Environments. Automated Generation does not migrate files unless they are in one of the locations associated with the **Initial Stage**. (Therefore, in order for Automated Generation to migrate a file to your production Environment, it must exist in your development Environment.)

The Maintain Versions window's **Actual Envs** field displays the Environments where the module currently exists. It is populated after Automated Generation runs. In order for this field to show accurate information for the Initial Stage Environment, your Codebase Administrator must run Automated Versioning (p. 91) prior to Automated Generation.

Promote, generate and install code

Follow the steps below to promote code from the Initial Stage Environment to another Environment.

1. Select your **Project** in the Maintain Versions window's (p. 36) **Quick Find** area.
2. For each version that you want to promote, click the **Promote to** dropdown and select the Environment to promote to (during the next Automated Generation run).

> Integra Codebase assumes that the versions to be promoted exist in the Initial Stage Environment.

3. Click  (**Action > Save**) to commit your action(s).

> To quickly view all versions that have been marked for promotion, click the **Promote to** column heading.

Migrate Application Object Library (AOL) setups

Follow the procedures below to migrate Oracle E-Business Suite 11i Application Object Library (AOL) setups.

Download the setup

In the Oracle E-Business Suite environment that you are migrating FROM, log into the responsibility **AM Change Manager User**.

1 Run your preferred download program to download the AOL setup you are migrating. Navigate to the **View Requests** form to verify that it runs successfully.

2 Find the downloaded data file in the `$AMCM_TOP\out` directory.

3 Wait for Integra Codebase's Automated Versioning (p. 91) to create a version of the downloaded data file. The new version should appear in the Codebase's Maintain Versions window (p. 36).

Create a link

4 In Integra Codebase, select your **Project** in the Maintain Versions window (p. 36).

5 Click **Link Modules**. The Link Modules window (p. 51) opens.

6 Select **Row...Insert Row** from the menu bar, and enter the following values:

Module Type	The downloaded file's extension.
Custom Module	The downloaded file's name (without extension).
Oracle Module	`TEMPLATE`

7 Click  (**Action > Save**).

8 Close the Link Modules window.

Promote the setup

9 Find the downloaded file's version in the Maintain Versions window.

10 Click the version's **Promote to** dropdown, and select the Environment to promote to.

11 Click  .

12 Wait for Automated Generation to promote the data file to the specified Environment.

Verify the promotion's results

13 In the Oracle E-Business Suite Environment that you are promoting TO, log into the responsibility **AM Change Manager User**.

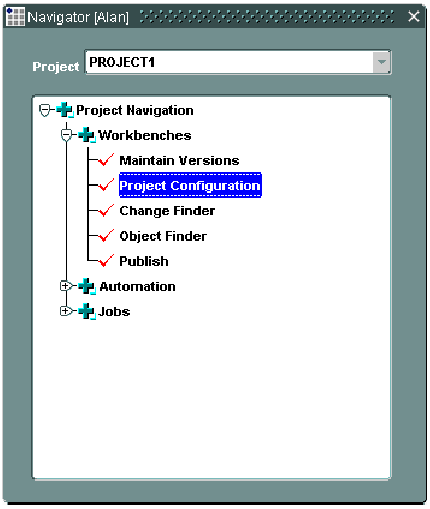
14 Wait for the completion of the concurrent request upload submitted by Automated Generation (p. 101).

> The name of this program begins with the word “Upload,” and ends with the name of the custom concurrent program, in parentheses.

15 ***If you have uploaded an Oracle E-Business Suite Key Flexfield:*** Compile the Key Flexfield within your Oracle Environment.

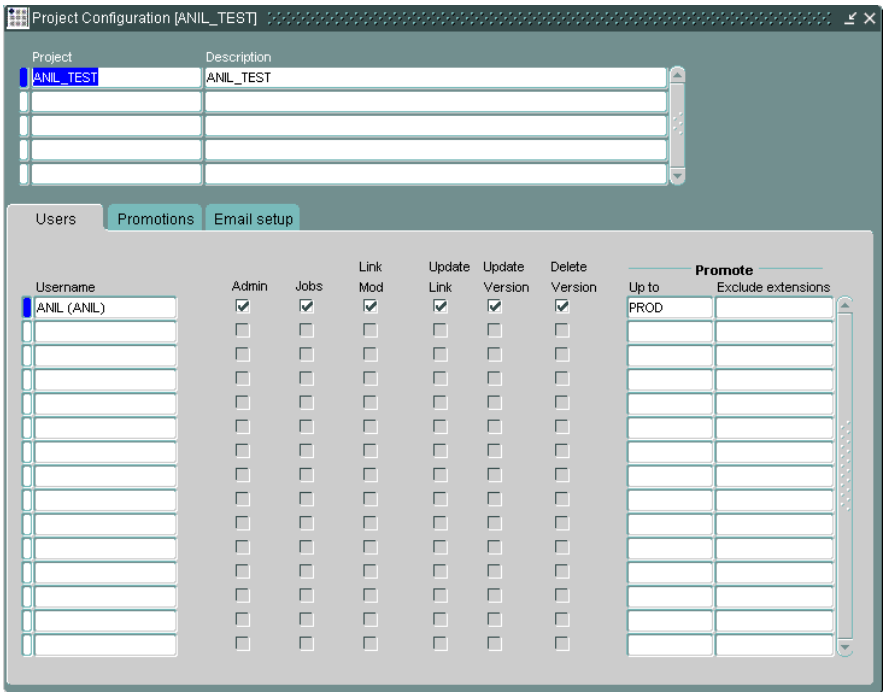
16 In Integra Codebase's Maintain Versions window, verify that the migrated version's **Actual Envs** field includes the destination Environment.

Project Configuration



This window lets you perform the same functions as the Configuration: Projects window (p. 25), but only for the project selected in the Navigator's Project field (p. 10).

Users



This tab lets you give Codebase users permission to use the project, and control what they can do with the project.

To add a user to a project:

1 Select the **Project** in the upper portion of the window.

2 Click on any **Username**, and then click  (**Row > Insert**). A blank row is highlighted.

3 In the blank row, select the **Username**, fill in the other fields, and click  (**Action > Save**).

To change a user's privilegeS:

1 Select the **Project** in the upper portion of the window.

2 Revise the user's fields, and click  .

To remove a user from a project:

1 Select the **Project** in the upper portion of the window.

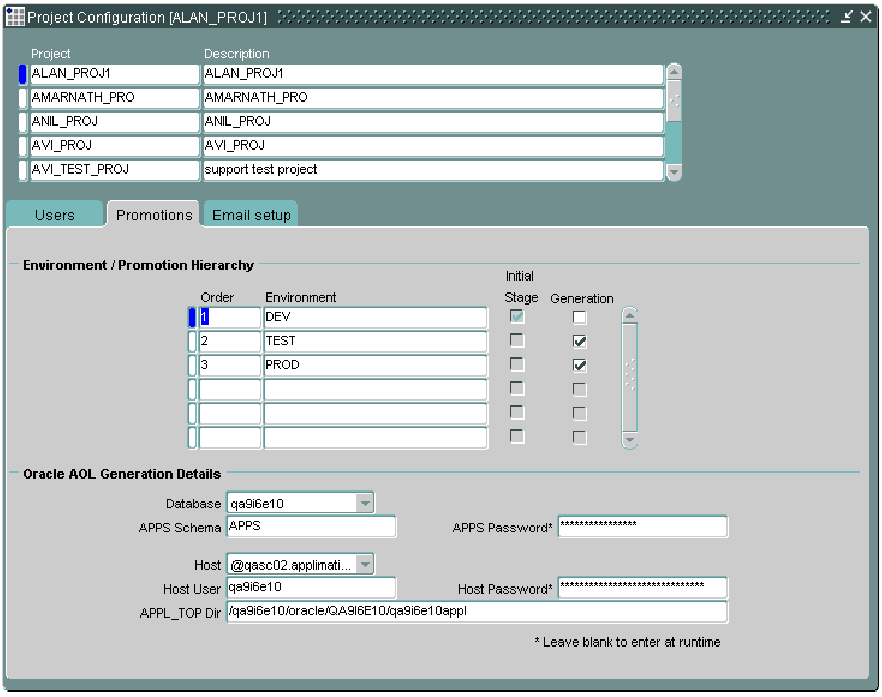
2 Click on the Username, and then click  (**Row > Delete**).

The tab contains the following fields and buttons:

Username	Select the user who will have permission to use the project.
Admin	Grants access to the Project Configuration (p. 57) window.

Jobs	Grants the ability to run the automated processes (by enabling the **Schedule** button on the **General** tabs of their windows). Consider creating a user with only the `Projects` role, and giving this user the sole ability to schedule all projects' automated processes. By limiting scheduling capability to a single user, you simplify auditability, and decrease the chance of scheduling conflicting or duplicative processes (potential conflicts are described in each process' documentation). Users are created on the Administration: Users tab (p. 17).
Link Mod	Grants the ability to create records in the Link Modules window (p. 51).
Update Link	Grants the ability to update records in the Link Modules window.
Update Version	Grants the ability to update records (e.g., versions) in the Maintain Versions window.
Delete Version	Grants the ability to delete versions (p. 49) in the Maintain Versions window.
Promote: Up to	Limits how far the user can promote modules (p. 54).
Promote: Exclude extensions	Comma-delimited list of file extensions that the user will NOT be able to promote.

Promotions



This tab lets you define the promotion hierarchy, which is followed when promoting code from your development Environment to other Environments (and eventually to your production Environment). Here is an example of a simple promotion hierarchy:

1. Development
2. Test
3. Production

> If your project will use Automated Versioning, but not the promotion mechanism, do not enter any records in this tab.

The **Initial Stage** checkbox indicates where all code will originate; only one of these checkboxes can be checked. The Initial Stage Environment should be the one where developers are creating code.

When defining the promotion hierarchy, the Environments must abide by the following rule: each non-initial-stage Environment (e.g., TEST, PROD) must correspond to an Initial Stage Environment (e.g., DEV) by containing matching Compilation Schemas (see Automated Generation: File Locations (p. 108)).

To define a project's promotion hierarchy and/or AOL details:

1 Select the **Project** in the upper portion of the window.

2 Set the tab's fields as desired, and click  (**Action > Save**).

The tab contains these fields and buttons:

Define Environment/Promotion Hierarchy

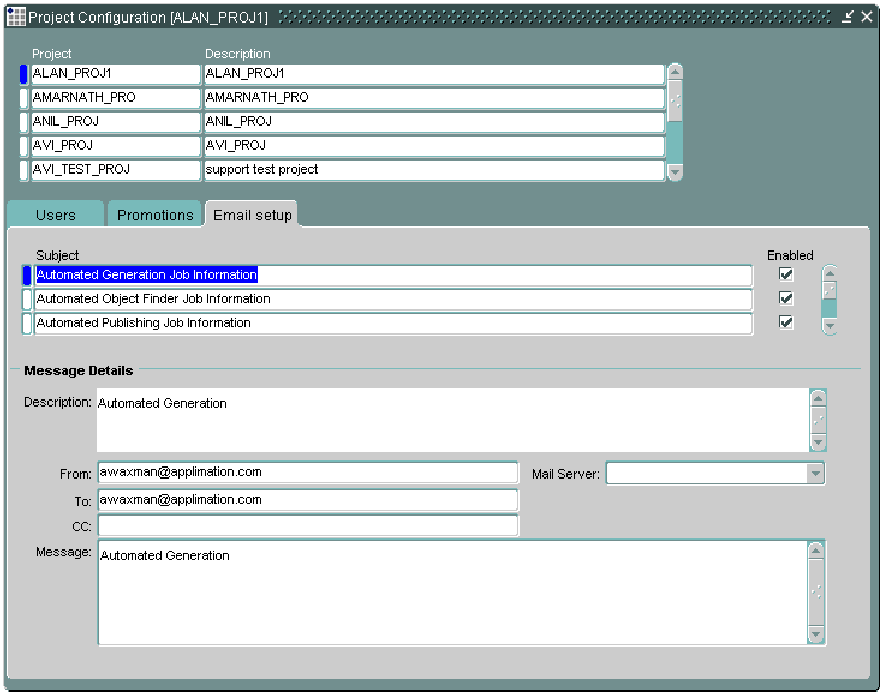
Order	Order in which code is promoted, starting with the Initial Stage (e.g., Development).
Environment	Name of the Environment.
Initial Stage	When checked, identifies the Environment that code originates from (i.e., the one where development occurs). See above for more information.
Generation	When checked, Automated Generation (p. 101) can perform all of its functions on the Environment. When unchecked, Automated Generation only promotes files; it does not run SQL or compile forms.

Oracle AOL Generation Details

Database	TNS entry on the Codebase Server for the Environment's database.
APPS Schema	Name of the Oracle E-Business Suite APPS schema (typically `APPS`).
APPS Password	Password of the APPS schema. Optional; if you do not enter a value, it will be requested at runtime.
Host	***Unix/Linux only:*** Host used to access the **Database**.

Host User	***Unix/Linux only:*** Username for logging into **Host** (typically `APPLMANAGER`). Enabled if **Host** is chosen.
Host Password	***Unix/Linux only:*** Password for logging in as **User**. Enabled if **Host** is chosen. Optional; if you do not enter a value, it will be requested at runtime.
APPL_TOP Dir	***Unix/Linux only:*** Full path on host to the `APPL_TOP` directory. Enabled if **Host** is chosen.

Email setup



This tab lets you configure the email messages generated by the project.

To create a message:

1 Select the **Project** in the upper portion of the window.

2 Click any **Subject** row, and then click (**Row > Insert**). A blank row is highlighted.

3 Fill in the blank row's fields, and click (**Action > Save**).

To edit a message:

1 Select the **Project** in the upper portion of the window.

2 Click the message's **Subject**.

3 Revise the message's fields, and click .

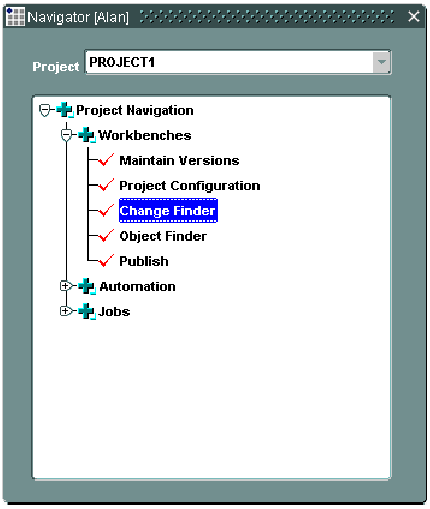
The tab contains these fields:

Subject	The message's "Subject" header.
Enabled	When checked, the message can be sent; when unchecked, the message cannot be sent.

Message Details

Description	Short description of the message. Appears only on this tab; does not appear in the message.
From	Usually the email address of the Project Administrator, so developers' replies will be sent back to the Administrator.
To	Email address(es) used when no address can be obtained for the developer associated with a file in the Link Modules (p. 51) or Maintain Versions (p. 36) windows. Separate multiple email addresses with commas.
Cc	Email address(es) that should get a copy of the message.
Mail Server	Outbound email server to use. Email servers are specified on the Administration window's Email Servers tab (p. 23).
Message	The message's body. All **Subject** variables (listed above) can be used in **Message**, as well as: `sub_comments` – **Comments** in Maintain Versions (p. 36) for the file being processed. `sub_obj_names` – List of objects containing PL/SQL code that are no longer inherited. This runtime variable is available only when Automated Publishing (p. 124) sends a warning message that PL/SQL code is no longer inherited.

Change Finder



Change Finder detects changes in forms, menus, reports, libraries, text files, database objects, and entire database schemas. It identifies all differences between files or objects, and displays them in easy-to-read reports.

Change Finder can be useful to nearly everyone in your information systems department. It is handy for resolving differences between Environments, and finding changes to a given form, report, or library.

DBAs use it to determine where database schemas are out of sync. This is valuable when your Test database needs to match your Production database, but for unknown reasons they don't. Change Finder can tell you all of the differences between the two schemas. It can also display files that contain PL/SQL that is no longer inherited, making it easy to determine whether any PL/SQL properties have been accidentally overwritten.

Operations personnel use it to determine if there is a problem in the production schema compared to another baseline schema (e.g., development, test, etc.).

Testers can view detailed differences between a Baseline (Original) Form/Report and a Changed (Customized) Form/Report. It also allows them to verify that their database schema matches the developer's database schema.

Code Review Teams can view detailed differences between a Baseline (Original) Form/Report and a Changed (Customized) Form/Report so the entire team does not have to walk through every line of code.

Programmers and Analysts can see the changes that have occurred in forms, reports, libraries, database objects, and schemas when compared to other Environments or versions.

Change Finder cannot run concurrently with itself, Automated Publishing, or Automated Change Finder.

Helpful hints

- When comparing one object or file to another, the names do not have to match; therefore, you can compare two procedures or procedures with different names.
- As with all Project windows, this window is monitored to ensure that the maximum concurrent user license is not exceeded. The concurrent user count is incremented each time this window is opened. The concurrent user count is decremented when you exit the window. If the window does not exit normally (e.g., computer turned off or crashed, or program aborted), the concurrent user count will not be decremented. The abnormal terminations will automatically be cleaned up the following day when someone exits Change Finder normally. In the meantime, excessive abnormal terminations of windows may keep developers from accessing windows until the following day. Simply entering the window and exiting again will start the cleanup process.
- Because of the cleanup process described above, you should never leave the Change Finder running overnight, because internal data will be deleted the next time someone exits the Change Finder.

Automated Change Finder (p. 115) calls Change Finder; thus, the nightly batch process could also delete your Change Finder data.

Troubleshooting

Change Finder informs you when it is unable to analyze files. When it does, use the following troubleshooting techniques:

Form, Menu or Library Comparison Problems

In the event that Change Finder cannot analyze an Oracle form, menu or library, follow these steps:

1. Open the file using Oracle Forms Builder. If you encounter errors when opening the file, resolve the errors and rerun the comparison.
2. Verify that the **Start In** directory for the Oracle Forms Builder icon is set to the `<ORACLE_HOME>/bin` directory. Attempt to open the file again using Oracle Forms Builder. If you encounter errors when opening the file, resolve the errors and rerun the comparison.
3. Look for error files (which have the extension **.err**) in the temporary directory specified in the General tab's (p. 73) **Temp Dir** field for your project. If error files exist:
    a. Open and review the error files.
    b. Resolve all errors.
    c. Rerun the comparison.

Report Comparison Problems

In the event that Change Finder cannot analyze an Oracle report, follow these steps:

1. Open the file using Report Builder. If you encounter errors when opening the file, resolve the errors and rerun the comparison.
2. Save the report to the database within Reports Builder. If you encounter errors when opening the file, resolve the errors and rerun the comparison.
3. Verify that the **Start In** directory for the Report Builder icon is set to the `<ORACLE_HOME>/bin` directory. Attempt to open the file again using Oracle Reports Builder. If you encounter errors when saving the report to the database, resolve the errors and rerun the comparison.

4 Look for error files (which have the extension **.err**) in the temporary directory specified in the General tab's (p. 73) **Temp Dir** field for your project. If error files exist:

a. Open and review the error files.

b. Resolve all errors.

c. Rerun the comparison.

Object types and properties supported when comparing schemas

Object Type	Modified Properties[1] and Source
DB Links (Private)	host, username (dba_db_links)
DB Links (Public)	host, username (dba_db_links) Public DB links are only analyzed when comparing SYSTEM schema.
Directory Objects	directory_path (dba_directories) Directories are always owned by SYS; thus, the SYS schema must be compared to see this type of object.
Functions	status (dba_objects), text[2] (dba_source)
Packages	status (dba_objects), text[2] (dba_source)
Procedures	status (dba_objects), text[2] (dba_source)
Grants	grantable (dba_tab_privs) Shows object privileges granted to users or PUBLIC. Granting READ on a Directory object will only appear when comparing the SYS schemas.
Indexes	table_name, status, index_type, and uniqueness (dba_indexes); locality, partitioning_type, subpartitioning_type (dba_part_indexes)

Indexed Columns	descend, column_position (dba_ind_columns) Indexed columns appear as children of an index. Column expressions will appear as a column. So UPPER(DEPT) would appear on the report as a column much like DEPT_ID would appear on the report.
Index Partitions	partition_position, high_value, tablespace_name, status (dba_ind_partitions)
Index Partition Key Columns	
Index Subpartitions	partition_position, tablespace_name, status (dba_ind_subpartitions)
Index Subpartition Key Columns	
Java Class	status (dba_objects)
Java Resource	status (dba_objects)
Java Source	status (dba_objects), text[2] (dba_source)
Library Objects	file_spec, dynamic, status (dba_libraries)
Materialized View Logs	rowids, filter_columns, primary_key (dba_snapshot_logs)
Operators	
Operator Bindings	function_name, implementation_type_schema, implementation_type, return_schema, return_type (dba_opbindings)
Operator Binding Arguments	argument_type (dba_oparguments)
Operator Ancillary Bindings	
Sequences	increment_by, min_value, max_value, order_flag, cycle_flag (dba_sequences)

Synonyms (Private)	table_owner, table_name, db_link (dba_synonyms)
Synonyms (Public)	table_owner, table_name, db_link (dba_synonyms) Public synonyms are only analyzed when comparing SYSTEM schema.
Tables and Materialized Views	temporary, iot_type, nested, row_movement, partitioned (dba_tables); partitioning_type, subpartitioning_type (dba_part_tables); compile_state, refresh_mode, refresh_method, build_mode, master_link, rewrite_enabled, rewrite_capability, fast_refreshable, last_refresh_type, after_fast_refresh, staleness, query, updatable (dba_mviews) Properties obtained from dba_part_tables will only appear when the **Include Partition Comparison** checkbox is checked. Properties obtained from dba_mviews will only appear for tables that are Materialized Views. The **query** property will appear under the "Modified Source" section of the report.
Table and Materialized View Columns	data_type, data_precision, data_scale, data_length, data_type_mod, data_default, nullable (dba_tab_columns)
Table and Materialized View Partitions	partition_position, high_value, tablespace_name (dba_tab_partitions)
Table and Materialized View Partition Key Columns	
Table and Materialized View Subpartitions	partition_position, tablespace_name (dba_tab_subpartitions)

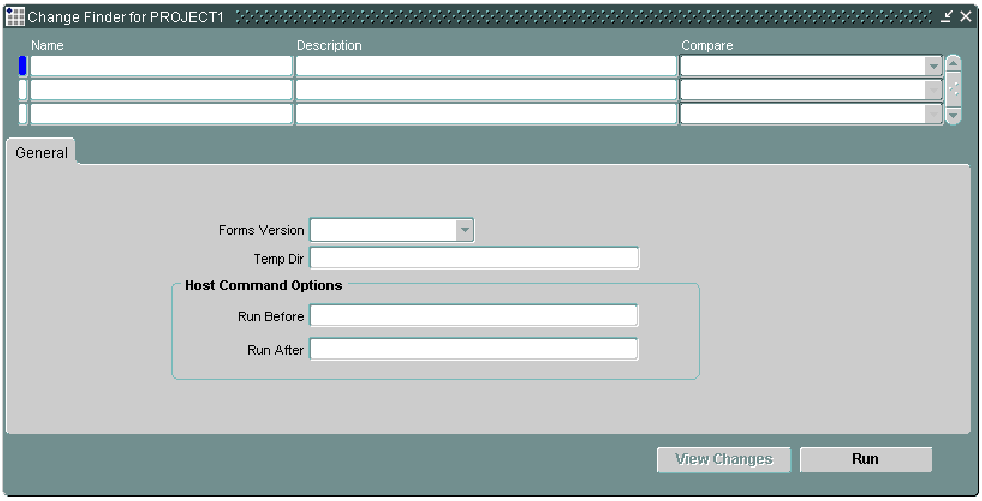
Table and Materialized View Subpartition Key Columns	
Table and Materialized View Constraints *(includes Primary Keys, Foreign Keys, Check Constraints, and Unique Constraints)*	status, delete_rule, r_owner, r_constraint_name, validated, rely, bad, deferred, deferrable, search_condition (dba_constraints) Constraints will appear on the report under their respective tables. Primary keys are matched based on existence, not name. This allows primary keys to have system generated names. However, foreign key, check, and unique constraints are match based on their name; thus, user-defined names should be used when defining these types of constraints. Using system generated names can cause the report to show a constraint as Deleted and New because it contains different system generated names.
Table Constraint Columns *(includes Primary Keys and Foreign Keys)*	column_name (dba_constraints) Column positions are used for matching so the property change could show a different column_name in position 1 (or position 2, 3, 4, etc.) Constraint columns will appear as Deleted or New on the report under its parent constraint.
Triggers	status (dba_triggers), status (dba_objects)
Types	status (dba_objects), text[2] (dba_source) "text" shown from dba_source will show the same information as Type Attributes but in a different format. Besides showing Type Attributes information, other pertinent information could also be displayed.
Type Attributes	attr_type_name, precision, scale, length, and attr_type_mod (dba_type_attrs)

Type Bodies	status (dba_objects), text[2] (dba_source)
Views	view_type_owner, type_text, oid_text (dba_views); status (dba_objects); "Check Option" and "Read Only" constraints (dba_constraints)
Views Columns	data_type, data_precision, data_scale, data_length, data_type_mod, nullable (dba_tab_columns)

[1] See Oracle Documentation for more information on Static Data Dictionary views and field definitions.

[2] The procedure source for this object (`dba_source.txt`) is not considered to be changed if the only lines that contain changes are those that begin with `$Header:`. This is intended to ignore the version number and timestamp that exists in all procedure source for the Oracle Applications; quite often, they change when no other changes exist in the procedure.

General



The upper portions of the page contains these fields:

Name	Name of stored Change Finder settings.
Description	Description of stored Change Finder settings.
Compare	Select either: **Compare Specific Modules** displays the Modules tab (p. 77). **Compare Time Periods** displays the Directories tab (p. 76).

The **General** tab contains the following fields and buttons:

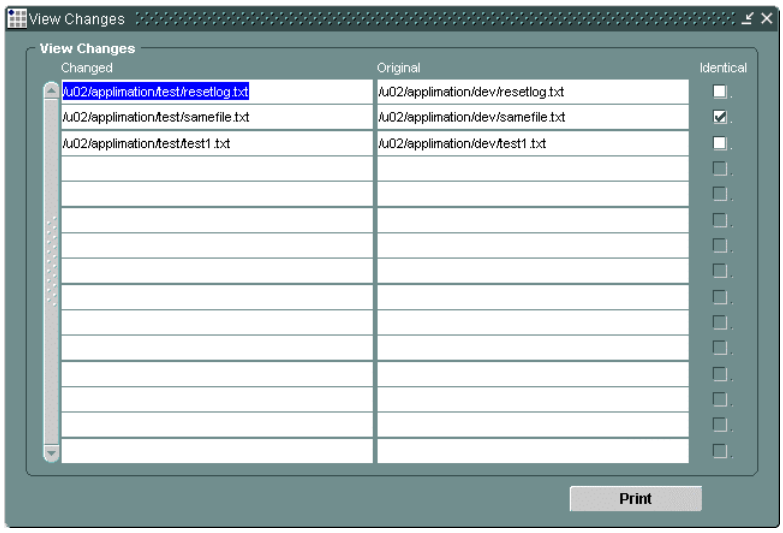
Forms Version	Enter a value to limit forms listed in search results to the specified version.
Temp Dir	Full path to a directory on Codebase Server where Codebase can temporarily store files.

Host Command Options

Run before	This command is used to run a program before Change Finder runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).
Run after	This command is used to run a program after Change Finder runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).

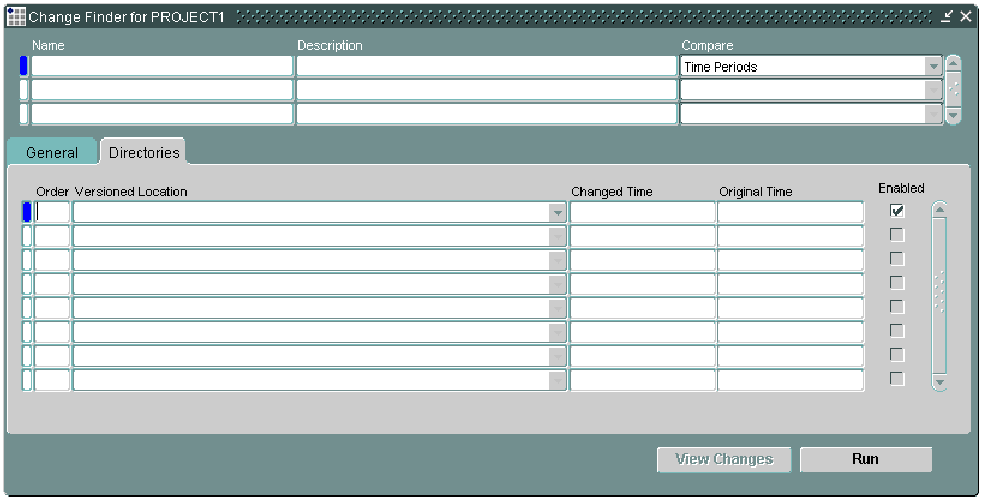
View Changes (button)	Displays a report of all changes.
Run (button)	Performs the search. When the search is complete, a report of all changes appears, and the **View Changes** button is enabled.

View Changes



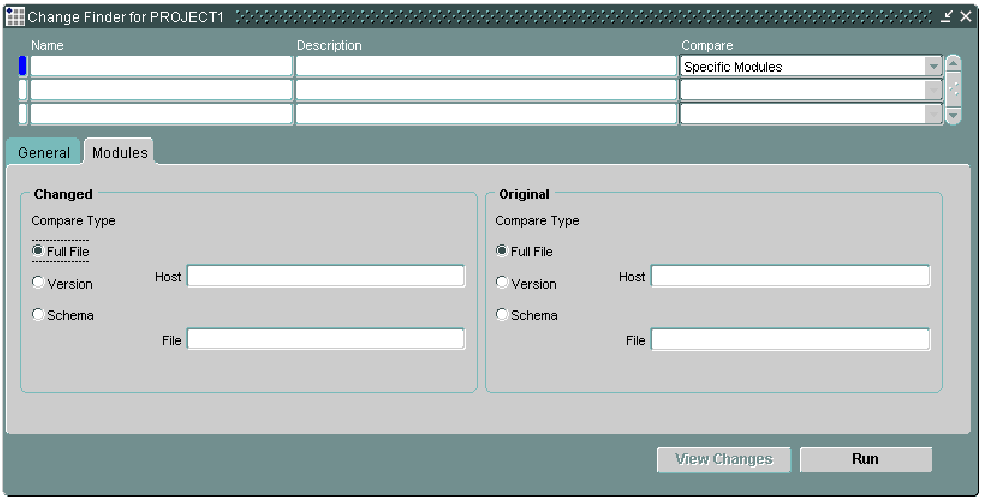
Lists all files that were tested; files marked **Identical** have not changed.

Directories



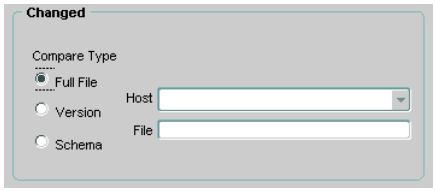
This tab appears when **Compare** is set to `Compare Time Periods`. It lets you compare how a directory's content has changed over time. Select one or more **Versioned Locations**, and for each, enter the **Changed Time** and **Original Time**. Codebase will compare the latest version modified before or at **Changed Time** to the latest version modified before or at **Original Time**. Enter the time in the format `MM/DD/YYYY hh24:mm:ss`

Modules



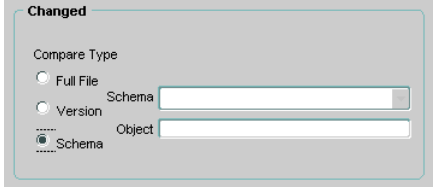
This tab appears when **Compare** is set to `Compare Specific Modules`. It lets you specify the content to compare. One each side of the tab (**Changed** and **Original**), select one of the following:

Full File - specify the **File**, and the **Host** used to access it.

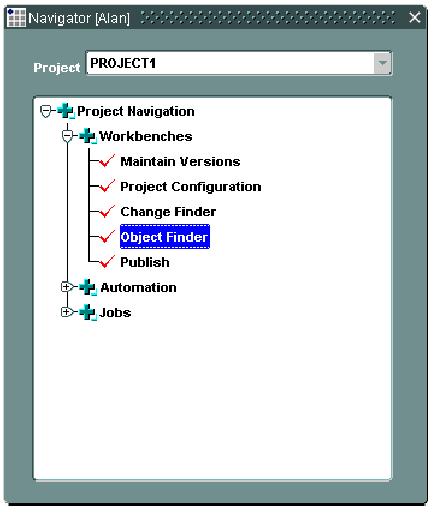


Version - choose a version to compare.

Schema - choose a **Schema** to compare, and optionally specify a single **Object** to compare.



Object Finder



Object Finder provides interactive, comprehensive impact analysis for scheduled code changes to forms, reports, menus, libraries and PL/SQL code. Using Object Finder, you can search for:

Procedures: Search for a text string in stored procedures

Files: Search for a text string in files (forms, reports, libraries, text files, etc.) for text strings

DB Dependencies: Search for database object dependencies

Subclassed Forms: Search for subclass relationships

Attached Libraries: Search for attachments between forms and libraries

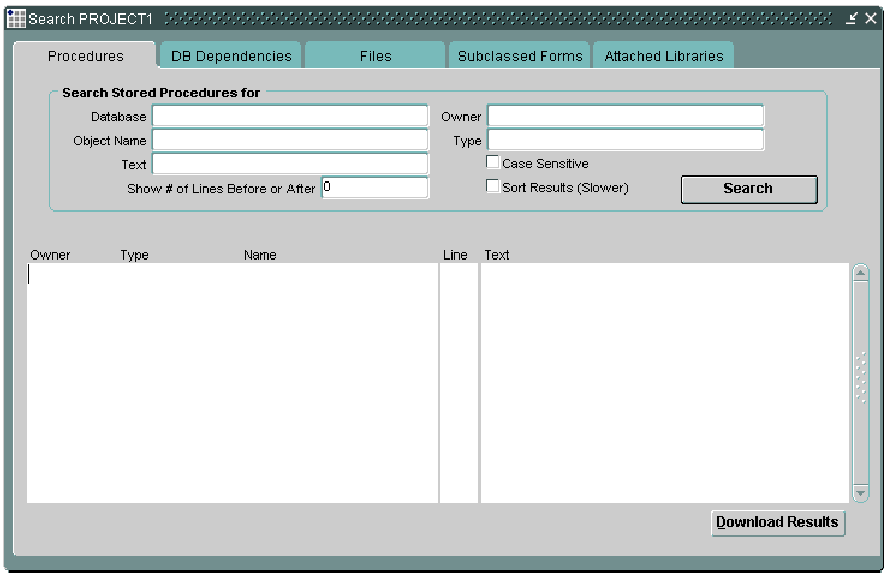
This window can assist you in performing impact analysis. Consider the following example: you are asked to modify a library but are concerned that it may cause other forms or libraries to work incorrectly. This screen allows you to see all of the forms and libraries that have this library attached to them. If no forms or libraries are found, you can then be assured that there will be no impact to other forms and libraries. On the other hand, if the query returns forms and libraries, you know that modifying this library could have an impact on them. Once you have made your changes, you might want to retest these forms and libraries to ensure that they still work correctly.

This window will also help you determine how many modules will be affected by a proposed customization. When planning a customization, query on the affected modules to get a list of modules that may also be affected. Using this screen while planning your customizations will increase the accuracy of your plans and budgets.

Do not run more than one session of Object Finder at once on your workstation.

The results of a query in the **Subclassed Forms** tab will return only those objects that are one hop away. Subclassing more than one hop away occurs when an object that is subclassed from Form A into Form B is again subclassed from Form B into Form C. In this case, entering a query for the object in Form A would NOT retrieve the object in Form C, because it is two hops away. If you need to find subclassed objects that are more than one hop away, refer to `\CMO\customer\objfnd.sql`. This script is capable of showing subclassed objects that are any number of hops away.

Procedures



This tab lets you find stored procedures that contain a specified text string. Optionally, you may specify additional criteria for limiting your query.

The information on this tab is always current, because it is extracted directly from your database. Because many records could match your query, they are fetched as you view/scroll them. This tab also lets you view an entire stored procedure by entering **Owner**, **Name** and **Type**.

To perform a search:

1 In the Navigator window (p. 10), select the **Project** to search.

2 Double click **Project Navigation > Workbenches > Object Finder**. The **Object Finder** window opens, with this tab (**Procedures**) displayed.

3 Select the database to search in the **Database** field.

4 Enter the information to search for (all fields are optional, but enter information in at least one):

Owner	Schema that owns the stored procedure.
Name	Stored procedure's name.
Type	Stored procedure's type (e.g., function, package, etc.). Click `Ctrl-L` to see a list of available procedure types.
Line	Line number shown in the text of the PL/SQL field.
PL/SQL	Text within one line of the stored procedure. For faster processing, also enter **Owner** and/or **Name** when specifying **PL/SQL**.

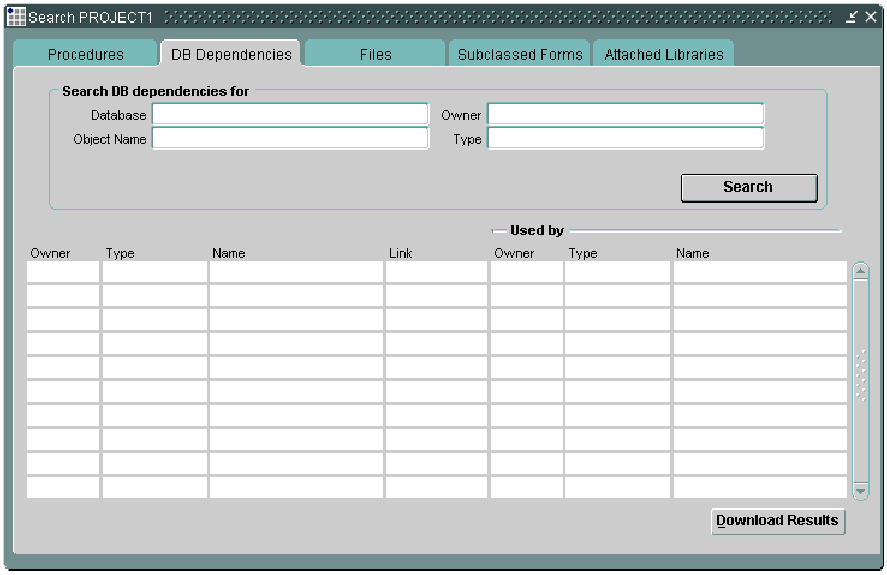
5 You can also choose either of these options:

Show # of Lines Before or After	Enter any positive integer to display the lines before and after each matching line.
Case sensitive	When checked, Object Finder matches capitalization exactly.

6 Click **Search**.

7 To save a report of your results, click **Download Results**.

DB Dependencies



This tab lets you find all database objects that subclass a specified object. The information on this tab is always current, because Object Finder extracts this information directly from your database.

To perform a search:

1 In the Navigator window (p. 10), select the **Project** to search.

2 Double click **Project Navigation > Workbenches > Object Finder**. The **Object Finder** window opens.

3 Click the **DB Dependencies** tab.

4 Enter the database name containing your objects in the Enter Database: field.

5 Select the database to search in the **Database** field.

6 Enter the information to search for (all fields are optional, but enter information in at least one):

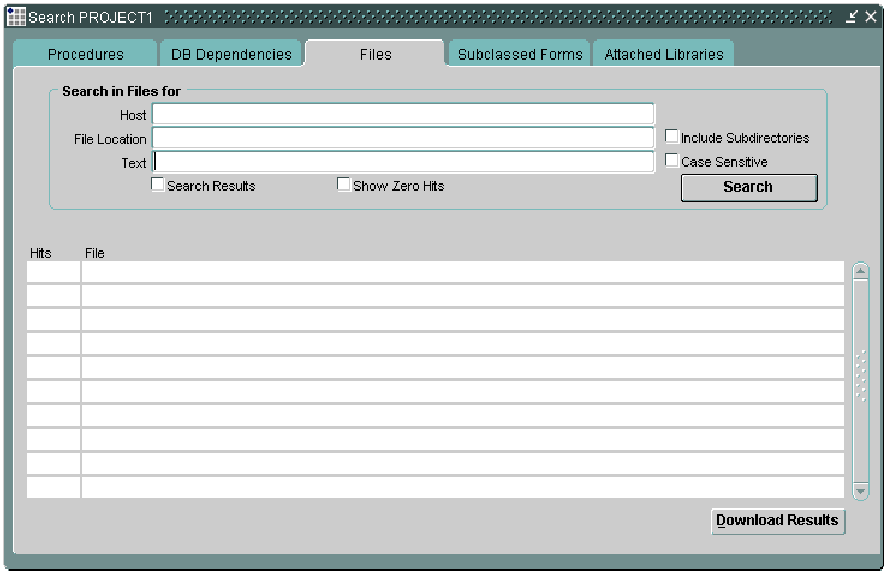
Owner	Schema that owns the stored procedure.
Type	Stored procedure's type. Click `Ctrl-L` to see a list of available procedure types.
Name	Stored procedure's name.

7 Click **Search**. The following information is displayed:

Owner	Schema that owns the referenced object
Type	Type of the referenced object
Name	Name of the referenced object
Link	Database link used to access the referenced object
Used By: Owner	Owner of the referencing object
Used By: Type	Type of the referencing object
Used By: Name	Name of the referencing object

8 To save a report of your results, click **Download Results**.

Files



This tab lets you find forms, reports, libraries and any other file types that contain a given text string. You can view any file listed in the results area by double-clicking its name in the **File** column.

To perform a search:

1 In the Navigator window (p. 10), select the **Project** to search.

2 Double click **Project Navigation > Workbenches > Object Finder**. The **Object Finder** window opens.

3 Click the **Files** tab.

4 Enter the text to find in the **Text** field. Regular expressions are accepted. Capitalization is ignored unless you check **Case sensitive**.

Take care when entering the characters shown below, which have special regular expression meanings in this field. To search for any of these characters, precede them with backslashes (`\`). For example, to search for `WHEN-NEW-FORM-INSTANCE`, enter `WHEN\-NEW\-FORM\-INSTANCE`.

`|` between expressions finds strings that match either the first expression OR the second expression. As many as ten expressions can be combined this way.

`^` as the first character of the pattern forces matches to beginnings of lines.

`$` as the last character of the pattern forces matches to ends of lines.

`.` anywhere in the string matches any single character.

`*` after an expression matches zero or more occurrences of that expression.

`+` after an expression matches one or more occurrences of that expression.

`-` after an expression optionally matches that expression.

`[`*characters*`]` matches any of the characters, but no others.

`[`*character1*`-`*character2*`]` matches the range of characters between *character1* and *character2.*

`[~`*characters*`]` matches any character except the characters in the brackets.

`\<` matches the beginning of a word.

`\>` matches the end of a word.

`\b` matches the backspace character (ASCII code 8).

`\n` matches the new line (or line feed) character (ASCII code 10).

`\f` matches the form-feed character (ASCII code 12).

`\r` matches the carriage return character (ASCII code 13).

`\t` matches the horizontal tab character (ASCII code 9).

`\`*character* matches the character, as explained above.

5 Enter the directory or filename that you want to search in the **Server Directory** field. You can specify wildcards using asterisks (`*`). Object Finder searches files with the extension **FMB**, **PLL** and **RDF** if you do not specify a file extension or use `*` in place of the extension.

To search PDF files, we recommend using Adobe Acrobat to ensure accurate results.

6 You can choose any or all of these options:

Include Subdirectories	When checked, also searches **Server Directory**'s subdirectories, and all descendent subdirectories.
Show Zero Hits	When checked, displays both the files that contain **Text** and those that do not. The **Hits** field will display `0` (zero) for all files that do not contain **Text**.
Case sensitive	When checked, matches the capitalization of **Text** exactly.

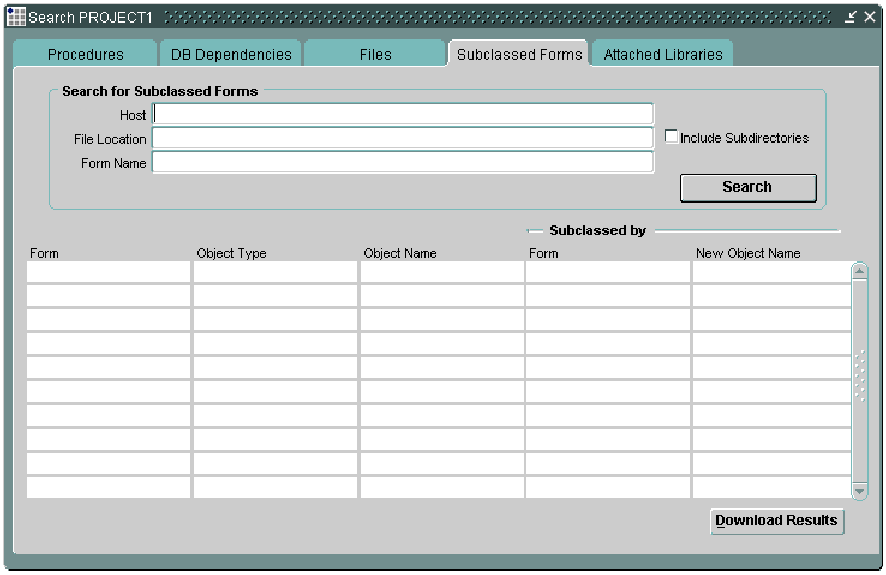
Search Results	When checked, searches the files already listed in the results area for the current **Text**. Use this option to find files that match two or more text strings.

7 Click **Search**. Object Finder responds with a message stating the number of files searched, and the number of files that contain **Text**. The results area shows the following information:

Hits	Displays the number of times that the text string was found in the file. Sometimes the number of **Hits** in a form or report is more than you can find when you perform a manual search. This is because Oracle sometimes leaves obsolete text in the binary files. To make the number of hits match, convert the binary file (`FMB`/`RDF`) to a text file (`FMT`/`REX`), and then back again. Rerunning the search will now give accurate results.
File	Displays the name and directory location of the file that contained the text string. These same results will be written to a tab-delimited file for printing and mailing purposes (file location will be shown at the end of your search). Double-click on this field to open the file using the associated application in your workstation's operating system. If no application is associated to this file, the file will not open.

8 To save a report of your results, click **Download Results**.

Subclassed Forms



This tab lets you find dependency information for forms subclassed by other forms, and/or forms that subclass other forms.

Your Codebase Administrator must run Automated Publishing (p. 124) to populate this tab with your Project's data. Automated Publishing should be scheduled to run nightly, to keep the information current.

This tab will only return objects that are one hop away. Subclassing more than one hop away occurs when an object that is subclassed from Form A into Form B is again subclassed from Form B into Form C. In this case, entering a query for the object in Form A would not retrieve the object in Form C, because it is two hops away. If you need to find subclassed objects that are more than one hop away, use `\CMO\customer\objfnd.sql`, which is capable of showing subclassed objects separated by any number of hops.

To perform a search:

1. In the Navigator window (p. 10), select the **Project** to search.
2. Double click **Project Navigation > Workbenches > Object Finder**. The **Object Finder** window opens.
3. Click the Subclassed Forms tab.
4. ***Unix Environments only:*** Select **Host**.
5. Set the directory to search in **File Location**.

6 Set the **Dependency Name** to the name of the object that depends on the object being searched for.

7 You can choose this option:

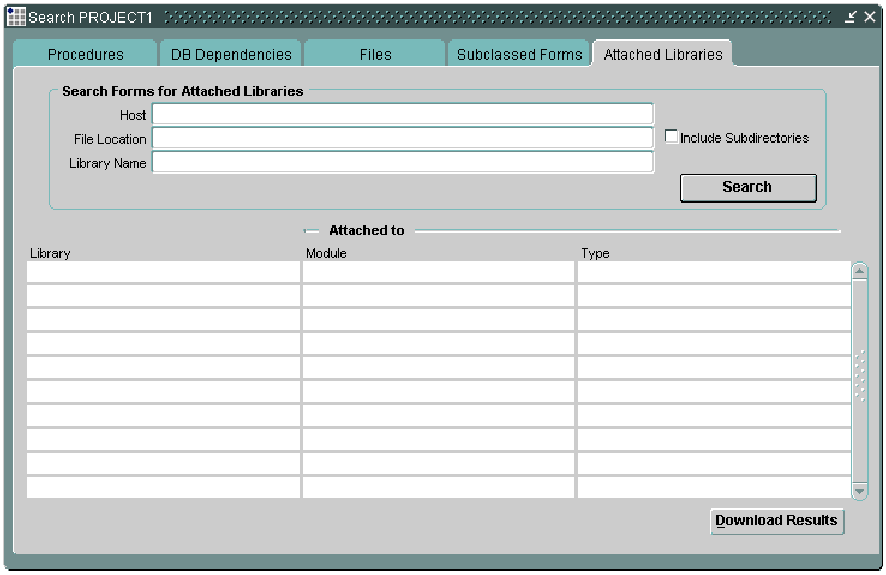
Include Subdirectories	When checked, also searches **Server Directory**'s subdirectories, and all descendent subdirectories.

8 Click **Search**. The results contain the following information:

Form	Name of the subclassed form.
Object Type	Type of the subclassed form.
Object Name	Name of the subclassed object - e.g., `Trigger Name`, `Alert Name`, `Block Name`, `Canvas Name`, `Editor Name`, `LOV Name`, or `Object Group Name`. For example, the Oracle E-Business Suite form `APPSTAND` contains a standard toolbar that most forms use. This toolbar's **Object Name** is `STANDARD_TOOLBAR`. Dot notation is used to represent child level objects. For example, `DEPTS.DEPT_NO.WHEN-VALIDATE-ITEM` describes the `WHEN-VALIDATE-ITEM` trigger on the item `DEPT_NO` that resides in block `DEPTS`.
Subclassed By: Form	Name of the form that subclasses the subclassed form.
Subclassed By: New Object Name	Name of the subclassed object within the subclassing form. After subclassing an object, its possible to change the name of the subclassed object. If the subclassed object was renamed, the new name will appear in this field.

9 To save a report of your results, click **Download Results**.

Attached Libraries



This tab lets you find all forms, libraries and menus that use a given library.

> Your Codebase Administrator must run Automated Publishing (p. 124) to populate this tab with your Project's data. Automated Publishing should be scheduled to run nightly, to keep the information current.

To perform a search:

1. In the Navigator window (p. 10), select the **Project** to search.
2. Double click **Project Navigation > Workbenches > Object Finder**. The **Object Finder** window opens.
3. Click the Attached Libraries tab.
4. ***Unix Environments only:*** Select **Host**.
5. Set the directory to search in **File Location**.
6. Set the **Library Name** to search for.
7. You can choose this option:

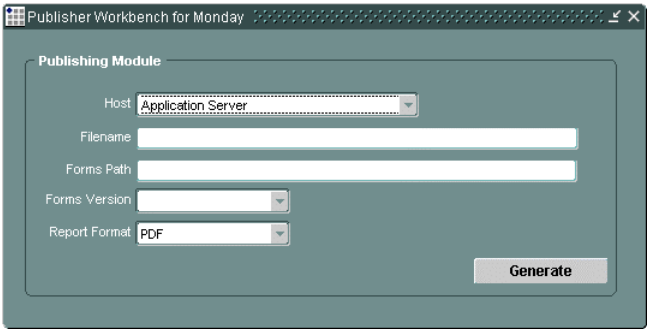
Include Subdirectories	When checked, searches **Server Directory**'s subdirectories, and all descendent subdirectories (as well as **Server Directory** itself).

8 Click **Search**. The results contain the following information:

Library	Name of the library.
Attached to: Module	Name of the form, library or menu using the **Library**. Opening this object in Oracle Forms Designer would show **Library** as an attached library. When querying on this field, you should enter a value for **Type** because a form and a library could have the same names.
Attached to: Type	Type of file identified in **Attached to: Module** (e.g., form, PL/SQL library, menu).

9 To save a report of your results, click **Download Results**.

Publishing



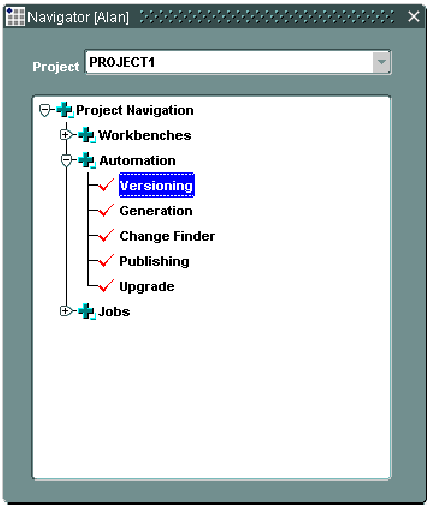
This window generates documentation of a single module (i.e., file or database object). It contains these fields and buttons:

Server	Server containing the module to be documented.
Filename	Full path and name of file to be documented.
Forms Path	Enter the Oracle Forms path environment variable (e.g., `FORMS90_PATH`) if **Filename** is a form (e.g., `FMB`, `PLL`, `MMB`).
Forms Version	Select an Oracle Forms version if **Filename** is an Oracle binary file (e.g., `FMB`, `PLL`, `MMB`).

Report Format	Select the report's format:
	`Delimited`
	`HTML`
	`PDF`
	`RTF`
	`XML`

Generate (button)	Displays the report in a new window.

Automated Versioning



Automated Versioning provides transparent version control by searching your application directories and databases, and automatically versioning changed files and database objects. It is a batch job that can be run manually on demand, or as a scheduled recurring job. The following modules (i.e., files and database objects) will be versioned:

- Files with the extension `FMB`, `INP`, `RDF`, or `PLL`
- File with extensions specified in the File Locations tab (p. 97)
- But not files specifically excluded on the Filters tab (p. 100)
- Database object types specified in the Schemas tab (p. 95)

Typically, you should run Automated Versioning on modules in your Initial Stage (i.e., development) Environment. If you want to use Automated Versioning to monitor other Environments, such as test or production, create a separate project for each Environment.

To use Automated Versioning:

As with all automated processes, you must have the Projects: Users tab's (p. 27) **Admin** privilege to configure and schedule this process, and the tab's **Jobs** privilege to view its execution status and logs.

1. In the Navigator window (p. 10), select your **Project**.
2. Double-click **Project Navigation > Configurations > Automated Versioning**.
3. Configure the window's tabs (General (p. 93), Schemas (p. 95), File Locations (p. 97) and Filters (p. 100)).
4. Click the General tab's (p. 93) **Schedule** button. The Schedule (p. 142) window appears.
5. Set all scheduling options as desired; you can run Automated Versioning once, or repeatedly. For example, you can version content every five minutes between 6am and 9pm.

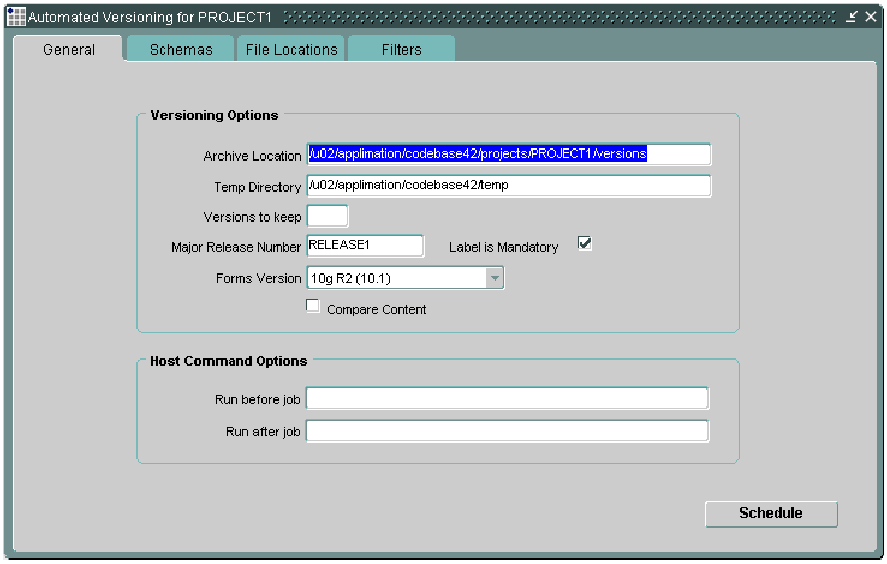
   If you are versioning database objects, the Environment's database must be running in order for Automated Versioning to function properly.

   You cannot run more than one instance of Automated Versioning at the same time, unless they are for separate projects. Automated Versioning can run concurrently with any other process.

6. Click the **Schedule** button. A new job is created; its activities can be monitored in the View Inside Jobs window (p. 136) window and its log (p. 140).

   When errors occur in the attempt to version a file because the file is open or locked, an error will be written to the log (p. 140). The program can be restarted at any time after a problem or system crash has occurred. The program will start processing where it left off.
7. After verifying that Automated Versioning has run successfully, view the Maintain Versions window (p. 36). This window lets you maintain, promote, and restore versions of a given program.
8. If you have scheduled repeated execution of Automated Versioning, check the log (p. 140) periodically to ensure that no errors occur.

General



This tab lets you configure Automated Versioning. All values entered in this tab will take effect after Automated Versioning is closed and restarted.

Versioning Options

Archive Location	This directory defaults to a directory created by Codebase on your Codebase Server when the project was created. It stores the archives of each module. Make sure there is plenty of disk space in this location. All files placed in this directory have the same name as the archived file plus an additional `.z` extension. Thus, the location specified must handle long file names. If you decide to change the default value, the directory must exist, be readable and writeable. Otherwise you will receive an error message.
Temp Directory	Full path on Codebase Server to directory where Codebase can temporarily store files.
Versions to Keep	Enter the number of versions of each file to keep, or leave blank to keep all versions.
Major Release Number	Use this value to distinguish between major releases. The value will be set by default, and will be displayed and editable in the **Maintain Versions** screen.

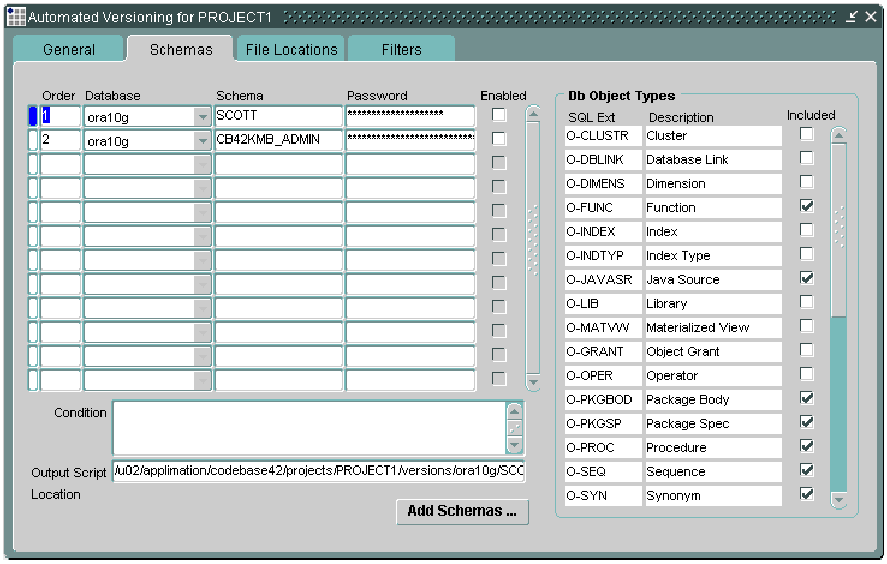
Label is Mandatory	When checking in a file (p. 47), the user must enter a value in the **Label** field.
Forms Version	The version of Oracle Forms used in the Environment. Used when **Compare Content** is checked.
Compare Content	When checked, Automated Versioning compares the actual contents of the module (i.e., file or object) rather than comparing only timestamps. Choosing this option can increase precision, but can also make Automated Versioning take dramatically longer.

Host Command Options

Run before	This command is used to run a program before Automated Versioning runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).
Run after	This command is used to run a program after Automated Versioning runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).

Schedule (button)	Begins the process of scheduling Automated Versioning. Displays the Schedule window (p. 142).

Schemas



This tab allows you to specify the schemas upon which to perform Object Versioning. Automated Versioning will first perform an extraction of DDL for the schema objects specified in this tab. The DDL will be placed into files using the extensions defined in the **Db Object Types** section. The files will then be versioned just like the other files that Automated Versioning processes. All values entered in this tab will take effect immediately (restarting Automated Versioning is not necessary).

Order	Enter a sequence number for the **Schema**. This sequence will determine the order in which the schemas are versioned. If an object with the same name exists in multiple schemas, the ordering is important because only the first occurrence of the object will be versioned.
Database	Enter the **Schema**'s database.
Schema	Enter the schema that you wish to version.
Password	Enter **Schema**'s password.
Enabled	When checked, objects in **Schema** are versioned.

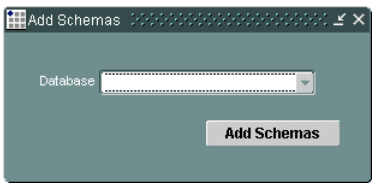
Condition	*Optional:* Either of these SQL WHERE clauses:
	`name_of_object comparison`
	`type_of_object comparison`
	For example:
	`name_of_object like '%ACCOUNT%'`
	If the condition generates an SQL error, Automated Versioning will fail.

Output Script Location	This field shows where the DDL scripts will be stored for each object. You may modify this directory but the location must be writeable. This field defaults to the **Archive Location** with the database and schema name appended.
Fill Schemas (button)	This button displays the Add Schemas window (p. 96).

Db Object Types

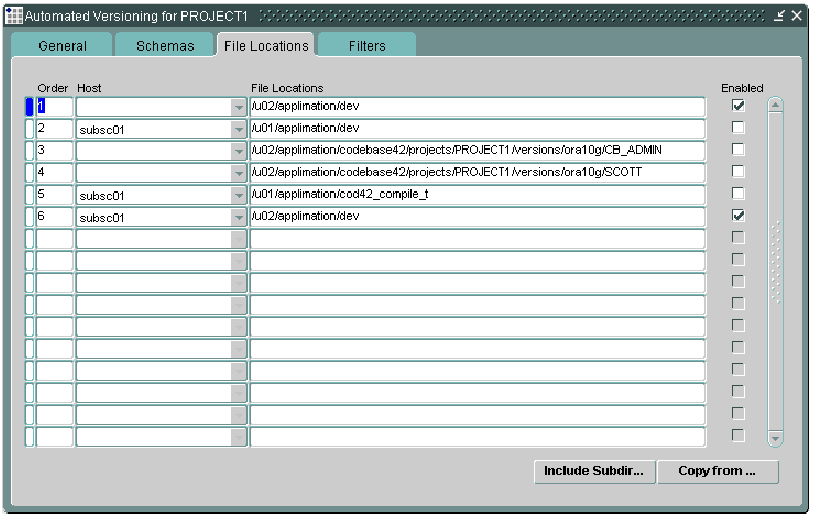
SQL Ext	List of known SQL object extensions.
Description	Object's name.
Included	When checked, this type of object is versioned.

Add Schemas



This window lets you add all of a database's schemas to the Schemas tab (p. 95). Select a **Database**, and click **Add Schemas**.

File Locations



This tab lets you specify directories to be searched for changed and new files. Automated Versioning finds files with the extension `FMB`, `PLL`, `INP`, `RDF`, or any other extension defined in the **File Types** area. All values entered in this tab take effect as soon as you click  (**Action > Save**).

> ***Oracle E-Business Suite users:*** Enter only those directories that contain custom modules. Automated Versioning will version all files in the specified directories, regardless of whether the file is linked as an Oracle module in Maintain Versions' Link Modules window (p. 51). This feature allows email messages to be sent automatically to the Codebase Administrator when a new custom file is introduced into your system.

If you have entered records in the Projects window's Promotions tab (p. 29), enter only file locations that correspond to an **Initial Stage**. Each file location must contain a unique set of file names, such that no file could be found in several file locations. The available List of Values (`Ctrl-L`) will help you restrict your entries to only those directories that have previously been entered for an Initial Stage Environment.

Fields

Order	Enter a sequence number for the **File Locations**. This determines the order in which locations are searched for files. The order becomes important when a duplicate file is found in several locations; the first occurrence is processed and other occurrences are skipped.

Host	***Unix/Linux only:*** Host used to access the **File Locations**.
File Locations	Enter valid directories that contain the files to be versioned. You may enter the directories with or without a backslash (\) at the end. Use the List of Values (`Ctrl-L`) to obtain file locations that have previously been entered in other components of the Codebase Suite. Locations specified in the Schemas tab's (p. 95) **Output Script Location** field are automatically added to this list. Checking or unchecking their **Enabled** boxes here does not affect the way Automated Versioning processes database schemas and objects - Automated Versioning always generates DDL files for the database objects specified on the Schemas tab, and places those files in the **Output Script Locations**. This tab's **Enabled** checkboxes merely determine whether those files will be versioned. You cannot query records in this Tab. The entire record set will always be displayed. To refresh this information, you can click on **Edit/Clear-Form** in the menu. The directory must be readable or you will receive an error.
Enabled	When checked, files in this location will be versioned.

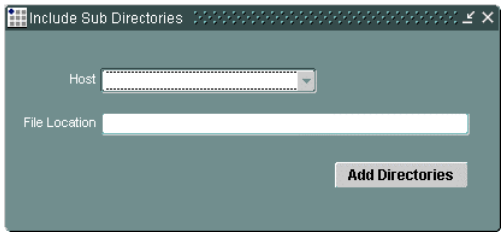
File Types

Extension	Files with these extensions will be versioned. `$$$` or `NONE` refers to files without extensions. To edit this list, use the Automated Generation window's Extensions tab (p. 111).

Text File	Indicates whether the file contains plain text.
LCT File	Indicates whether the file contains LCT content.

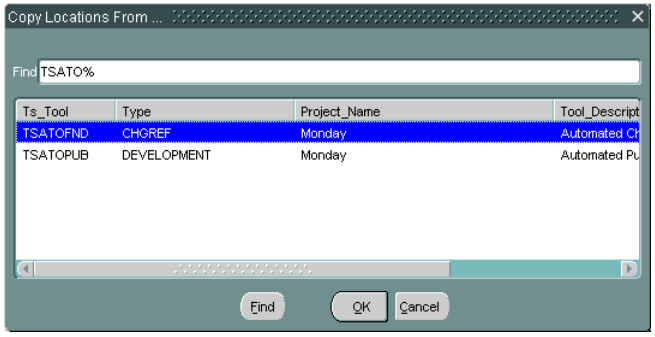
Insert Subdir... (button)	Displays the Include Subdirectories window (p. 99).
Copy From... (button)	Displays the Copy Locations From... window (p. 99).

Include Subdirectories



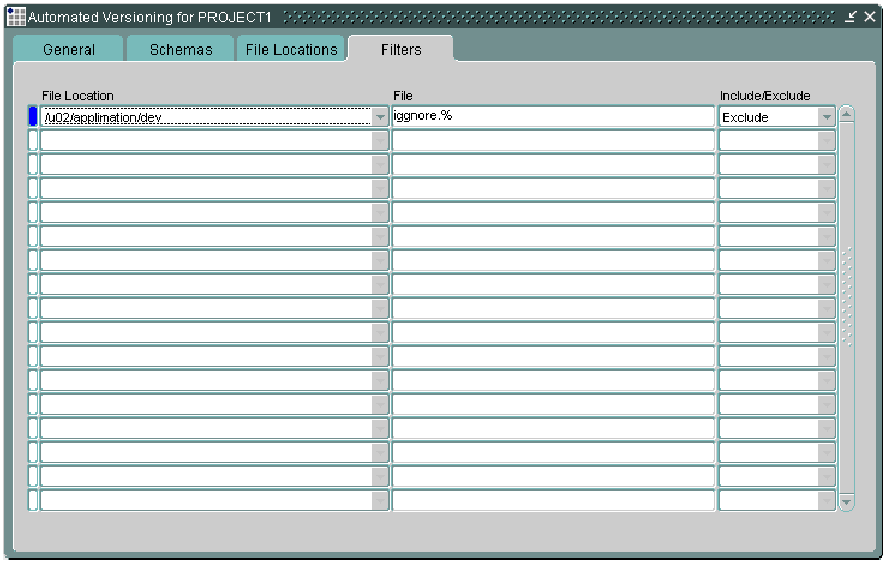
To add **File Locations** rows for each subdirectory of a specified directory, choose the appropriate **Host**, enter the full path to the parent directory in **File Location**, and click **Add Directories**.

Copy Locations From



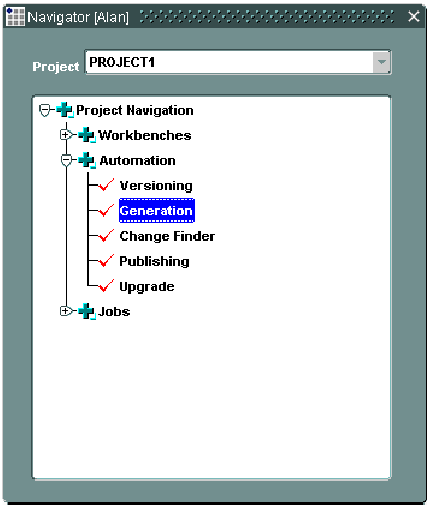
Select a location from the list and click OK. To filter the list, enter a value in the **Find** field (use % as a wildcard) and click the **Find** button.

Filters



To force the processing of specific files, select their **File Location**s, enter their names in the **File** fields, and set Include/Exclude as desired. You may use asterisks (*) as wildcards in the **File** field.

Automated Generation



Automated Generation performs code migration, installation and generation of software. It generates forms, reports, and libraries after running SQL scripts against a given database. It can also extract promoted files from the Automated Versioning archives, if you use Automated Versioning (p. 91) for version control.

Automated Generation is also capable of creating executables for any development tool (e.g., Powerbuilder, Visual Basic, etc.) by using the Extensions tab (p. 111) to specify the command to execute for a given file type.

Automated Generation automatically regenerates forms that reference changed forms.

When at least one schema is entered for an Environment, Automated Generation recompiles all invalid objects.

To use Automated Generation:

> As with all automated processes, you must have the Projects: Users tab's (p. 27) **Admin** privilege to configure and schedule this process, and the tab's **Jobs** privilege to view its execution status and logs.

1. In the Navigator window (p. 10), select your **Project**.
2. Double-click **Project Navigation > Configurations > Automated Generation**. The Automated Generation window appears.

3 At the top of the window, choose the destination **Environment** (i.e., the Environment to migrate *to*).

4 Configure the window's tabs (General (p. 104), Reports (p. 106), Schemas (p. 107), File Locations (p. 108), Filters (p. 110), Extensions (p. 111) and SQL Errors (p. 113)).

If you plan to use Automated Generation to migrate files, the directory structures in the source and destination Environments must match.

If you plan to use Automated Generation to generate SQL scripts, the scripts:

Must not use the `SPOOL` command. Integra Codebase logs script results itself.

Must not use the `EXIT` command, or Integra Codebase could be prevented from creating its log files.

Must not use the `CONNECT` command. All scripts must run against the compilation schema defined for the Environment.

Must not prompt for input. For example, do not use `&` or `&&`.

Oracle E-Business Suite users: We recommend that you configure Automated Generation to process all custom schemas within an Environment prior to processing the `Automated PublishingPS` schema, if it is present.

5 Click the General tab's (p. 93) **Schedule** button. The Schedule window (p. 142) appears.

6 Set all scheduling options as desired; you can run Automated Generation once, or repeatedly. For example, you can generate modules (i.e., files and database objects) nightly.

If you are generating database objects, the source and destination Environments' databases must be running in order for Automated Generation to function properly.

Do not run more than one instance of Automated Generation at the same time. Do not run Automated Generation concurrently with any process other than Automated Versioning. (You may run Automated Generation and Automated Versioning concurrently.)

7 Click the **Schedule** button. A new job is created; its activities can be monitored in the View Inside Jobs window (p. 136) and its log (p. 140).

When errors occur, they will be written to the log (p. 140). The program can be restarted at anytime after a problem or system crash has occurred. The program will start processing where it left off.

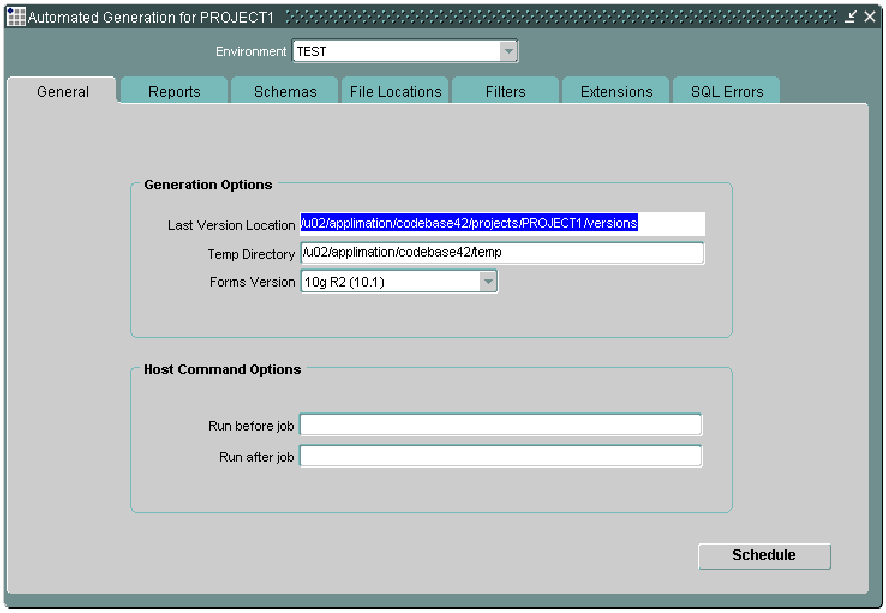
If Automated Generation fails to generate a file, it automatically attempts to regenerate the file on the next run. Likewise, SQL scripts will also be rerun if errors occurred on the previous run.

> Integra Codebase assumes that reprocessing modules will not harm your Environments. Your standards and Environments should reflect this assumption; for example, do not allow developers to use `DROP TABLE` or `DROP SEQUENCE` statements in any SQL scripts, so there's never a risk of accidentally dropping a table or sequence. (This is also true when using manual generation processes, because of the possibility of human error.)

When Automated Generation finishes, it sends the message Automated Generation Job Information to the recipients specified in the Projects window's Email setup tab (p. 32).

8 If you have scheduled repeated execution of Automated Generation, check the log (p. 140) periodically to ensure that no errors occur.

General



This tab contains the following fields and buttons:

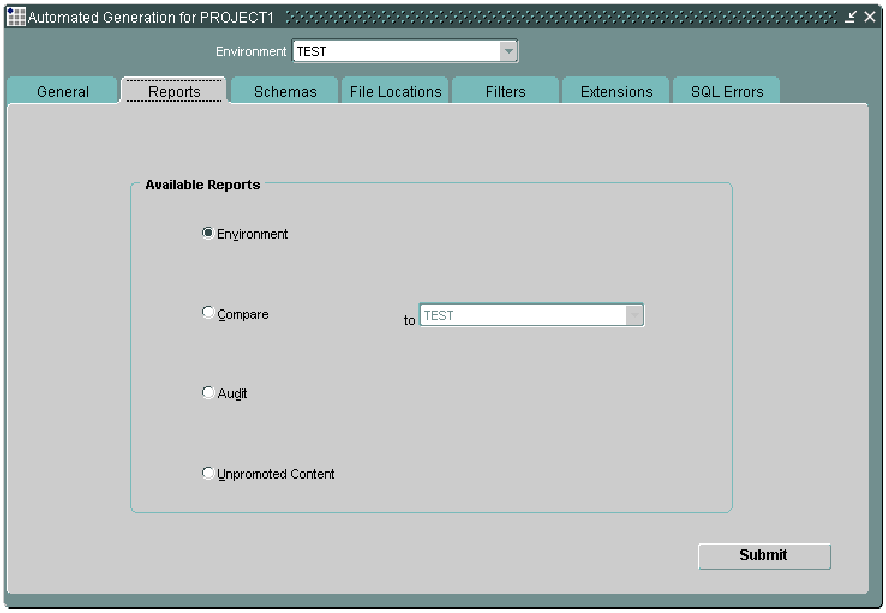
Generation Options

Last Version Location	This directory stores the last version of each module. It will be defaulted to a directory created by Codebase on your Codebase Server at the time the project was created. Make sure there is plenty of disk space for one version of each module. This directory must be different than the locations specified elsewhere in Codebase. The directory must exist, and be readable and writeable.
Temp Directory	Full path on Codebase Server to directory where Codebase can store files temporarily.
Forms Version	The Environment's Oracle Forms version (used to compile forms).

Host Command Options

Run before	This command is used to run a program before Automated Generation runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).
Run after	This command is used to run a program after Automated Generation runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).
Schedule (button)	Begins the process of scheduling Automated Generation. Displays the Schedule window (p. 142).

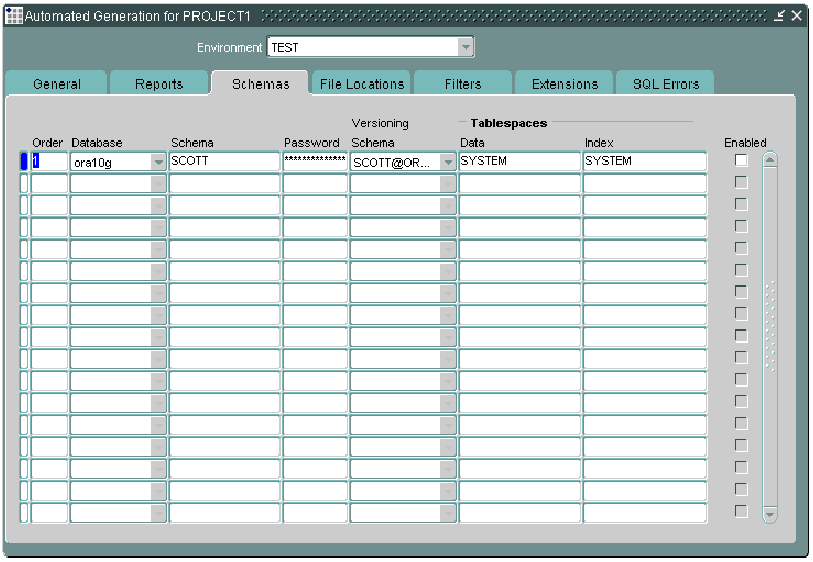
Reports



Four reports are available on this tab:

Environment	Shows modules (i.e., files and database objects) processed by Automated Generation on **Environment**.
Compare **to (dropdown)**	Compares the modules processed by Automated Generation on **Environment** to the environment specified in the **to** dropdown.
Audit	Shows differences between the version information stored in Codebase and the actual **Environment**.
Unpromoted Content	Shows modules in the Initial Stage Environment that were not promoted to **Environment**.
Submit (button)	Click this button to generate the selected report.

Schemas



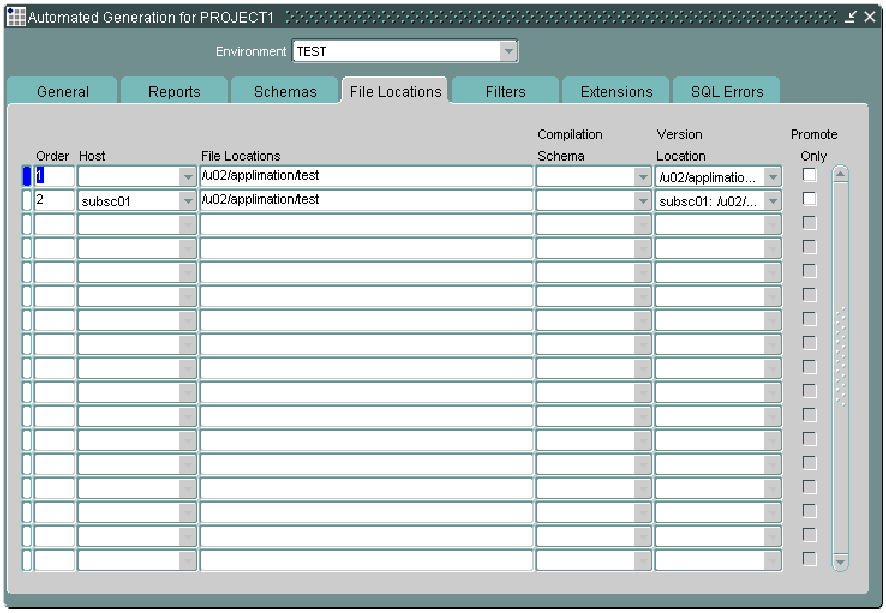
This tab contains these fields:

Order	Enter a sequence number for the **Schema**. This sequence will determine the order in which the schemas are processed. If an object with the same name exists in multiple schemas, the ordering is important because only the first occurrence of the object will be versioned.
Database	Select the **Schema**'s database.
Schema	Enter the schema that you wish to process.
Password	Enter **Schema**'s password.
Versioning Schema	The schema that **Schema** will be compared to. The schemas listed are specified on Automated Versioning's Schemas tab (p. 95).

Tablespaces

Data	Tablespace used to temporarily hold data while processing the schema.
Index	Index used to reference **Data**.
Enabled	When checked, the **Schema** is processed.

File Locations

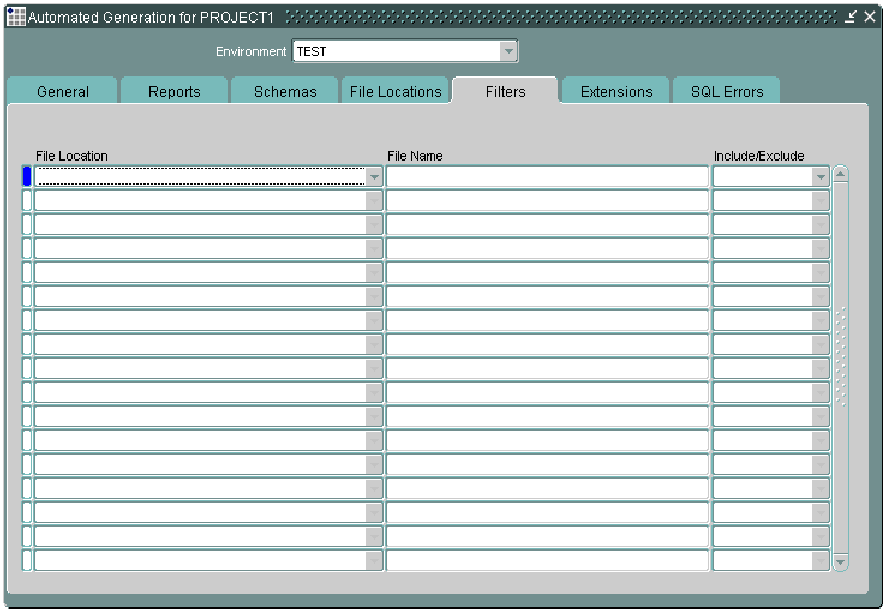


This tab contains these fields:

Order	Enter a sequence number for the **File Locations**. This sequence determines the order in which locations are searched for files. The order becomes important when a duplicate file is found in several locations; the first occurrence is processed and other occurrences are skipped.
Host	***Unix/Linux only:*** Host used to access the **File Locations**.
File Locations	Enter valid directories that contain the files to be versioned. You may enter the directories with or without a backslash (\) at the end. Use the List of Values to obtain file locations that have previously been entered in other components of the Codebase Suite. You cannot query records in this tab. The entire record set will always be displayed. To refresh this information, click on **Edit/Clear-Form** in the menu. The directory must be readable or you will receive an error.
Compilation Schema	Schema to use when compiling forms (to resolve references to database objects).

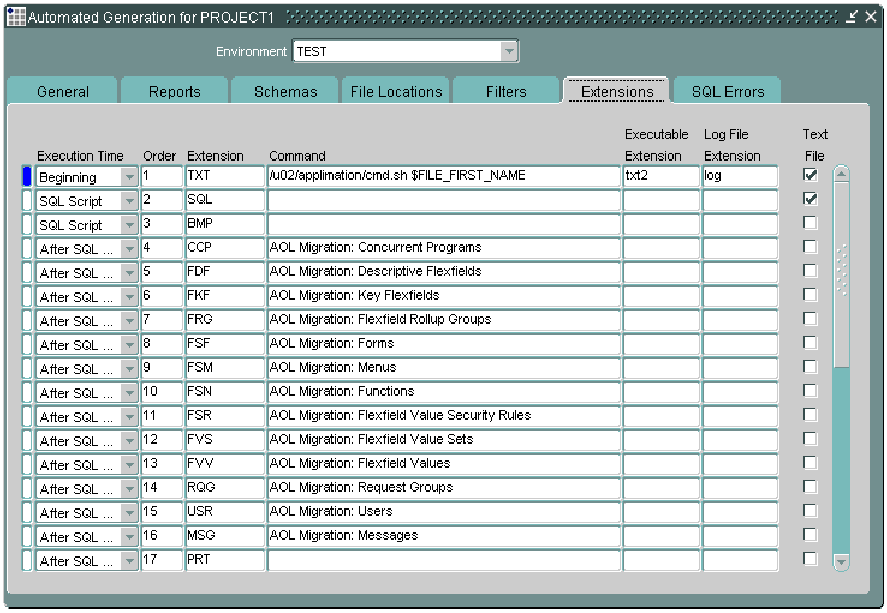
Version Location	Location containing the files that **File Locations** will be compared to.
Promote Only	When checked, Automated Generation promotes the file(s) without compiling them.

Filters



To force the processing of specific files, select their **File Location**s, enter their names in the **File** fields, and set Include/Exclude as desired. You may use asterisks (*) as wildcards in the **File** field.

Extensions



This tab specifies the file extensions that Automated Generation will process, how each extension will be processed, and the order in which they will be processed (e.g., a table must exist before an index can be created).

By default, files with the following extensions are ignored during automated versioning: wft, fmb, fmx, plx. For files with these extensions to be versioned, the extensions must be entered in this Extensions tab.

Execution Time This field, in combination with **Order**, determines the order in which the extensions are processed.

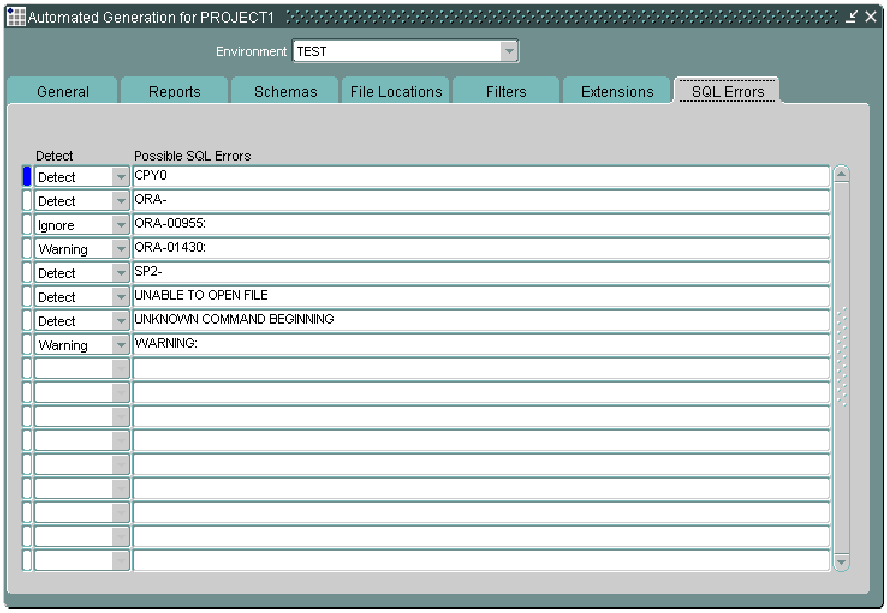
Extensions are processed in this order:

1. `Beginning`
2. `SQL Script`
3. `After SQL Scripts`
4. `End`

...and within each of those, by **Order**. Extensions with **Execution Time** set to `Never` are not processed.

Order	Enter a sequence number for **Extension**. This sequence, in combination with **Execution Time**, determines the order in which the extensions are processed.
Extension	Enter a filename extension. For files without an extension, enter $$$ or NONE as the extension.
Command	Operating system command to run after promoting the file. Typically used to compile the file's content (forms and SQL are compiled automatically). If the command fails, generation will not complete.
Executable Extension	If specified, Codebase will verify that a file with the versioned file's name and the specified extension was created.
Log File Extension	If specified, Codebase will search for a file with the versioned file's name and the specified extension, and if found, will add the file's contents to the log (p. 140).
Text File	If checked, Automated Publishing (p. 124) can document the file.

SQL Errors



This tab defines errors that should be detected in the SQL log files that are produced when Automated Generation runs SQL scripts. The log files for each SQL script will be placed in the same directory as the script itself.

Upon installation, default records will be created for some commonly occurring errors (as shown above). The errors that you define in this tab will depend on your standards and how SQL scripts are written. Because organizations handle error checking differently, additional errors/warnings can be added to this screen. Viewing the SQL error summary file could reveal errors that you might want to add.

> Until you are familiar with the Automated Generation process, do not remove any of the default records that appear on this tab.

Each SQL log file will be searched line by line for the error strings entered in this screen. The search is not case-sensitive. Enter one error per line; the percent symbol (%) is a wildcard representing zero or more characters.

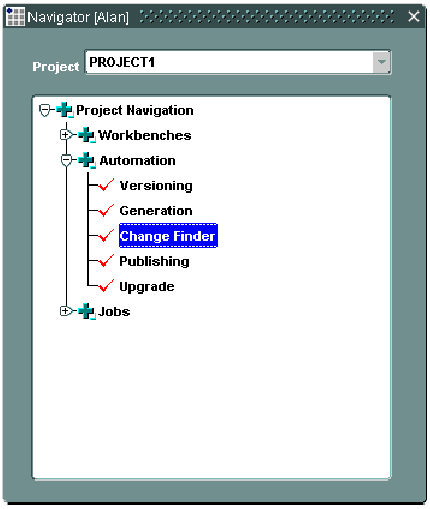
If any `Detect` error strings that do not match any `Ignore` or `Warning` error strings are found, the SQL script execution is considered a failure; thus, the SQL script will be rerun the next time that Automated Generation runs. If all `Detect` error strings match `Warning` or `Ignore` error strings, and at least one matches a `Warning` error string, the SQL script execution is considered successful and a notification is sent (see Projects: Email setup (p. 32)). If all `Detect` error strings match an `Ignore` error string, the SQL script execution is also considered successful.

> Regardless of the number of matching error strings for a `Warning` that are found, a notification will not be sent unless one of the `Warning` error strings also matches a `Detect` error string.

The tab contains these fields:

Detect	Determines whether to detect an error, ignore an error, or define a warning to be detected when SQL scripts are run. This is a dropdown list where you can choose `Detect`, `Ignore`, or `Warning`. For example, you may want to detect all `ORA-%` errors but choose to make all `ORA-00955` errors a warning. When a warning occurs, all processing continues as normal with the exception of the notification being sent. If you want to define a `Warning` error string, the same error string must match a `Detect` error string. If the `Warning` error string does not match a `Detect` error string, the string will be ignored (see above). Detecting `ORA-%` as an error is considered to include the warning `ORA-00955%` because of the wildcard (`%`).
Possible SQL Errors	Enter the error text to search for when Automated Generation runs SQL scripts.

Automated Change Finder



Automated Change Finder automates the process of creating difference reports for forms (`FMB` and `INP`), reports, libraries, text files, and database schemas. To keep this information up-to-date, we recommend that you run this process nightly.

Automated Change Finder, like Automated Publishing, processes files with any extension listed on the Automated Generation window's Extensions tab (p. 111).

Before you can use Automated Change Finder, you must add all schemas to be monitored for changes (on the Schemas tab (p. 120)), and add the locations of the form, report, library, and other files to be monitored (on the Directories tab (p. 121)).

To monitor *files* in more than one Environment (e.g., development, test, and production), create a project for each. You can, however, monitor your development, test, and production *database schemas* from within the same project.

> If a subclassed form is modified, and is being used in another form, the latter form will be compared to its previous version. Thus, the modification to the subclassed form will appear in the difference report (assuming the modified object was being referenced).

To use Automated Change Finder:

> As with all automated processes, you must have the Projects: Users tab's (p. 27) **Admin** privilege to configure and schedule this process, and the tab's **Jobs** privilege to view its execution status and logs.

1. In the Navigator window (p. 10), select your **Project**.
2. Double-click **Project Navigation > Configurations > Automated Change Finder**.
3. Configure the window's tabs (General (p. 117), Schemas (p. 120) and Directories (p. 121)).
4. Click the General tab's (p. 117) **Schedule** button. The Schedule window (p. 142) appears.
5. Set all scheduling options as desired; you can run Automated Change Finder once, or repeatedly. For example, you can publish documentation nightly to provide developers with up-to-date information.

   > Never run more than one instance of Automated Change Finder at the same time, unless the process is for separate projects. Do not run Automated Change Finder and Change Finder concurrently.

6. Click the **Schedule** button. A new job is created; its activities can be monitored in the View Inside Jobs window (p. 136) and its log (p. 140).

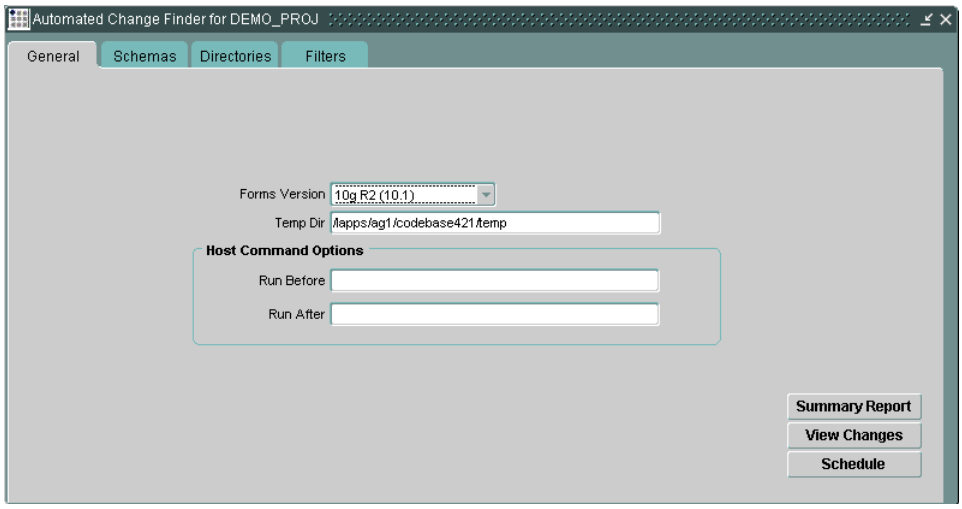
   When errors occur, they will be written to the log (p. 140). The program can be restarted at anytime after a problem or system crash has occurred. The program will start processing where it left off.
7. After verifying that Automated Change Finder has run successfully, you can view the results by clicking the General tab's View Changes button (p. 75).
8. If you have scheduled repeated execution of Automated Change Finder, check the log (p. 140) periodically to ensure that no errors occur.

To compare directory contents:

It is possible to compare two directories, but you'll need to create another project so you don't erase existing Automated Change Finder data. After creating another project, follow the instructions below to perform a comparison of multiple directory structures:

1. Enter all directory structures in the Directories tab (p. 121) for the new project.
2. Run Automated Change Finder to establish a baseline.
3. Either change the directories that were entered in Step 1, or remap your network drive to point to the new directory structures.
4. Rerun the Automated Change Finder to capture all the changes.
5. Click the General tab's (p. 117) **View Changes** button to review the differences.

General



This tab contains the following fields and buttons:

Forms Version The version of Oracle Forms installed in the project's Environments.

Temp Dir Full path on the Codebase Server to a directory where Codebase can temporarily store files.

Host Command Options

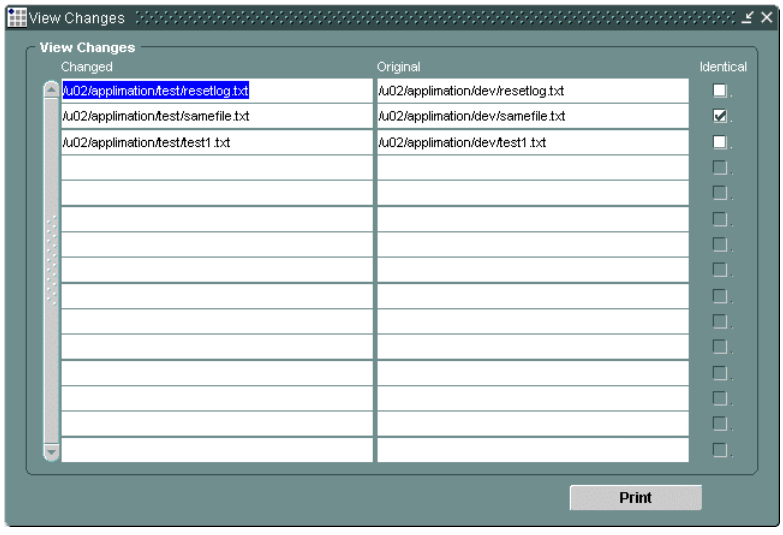
Run before This command is used to run a program before Automated Change Finder runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here.

The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string.

The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).

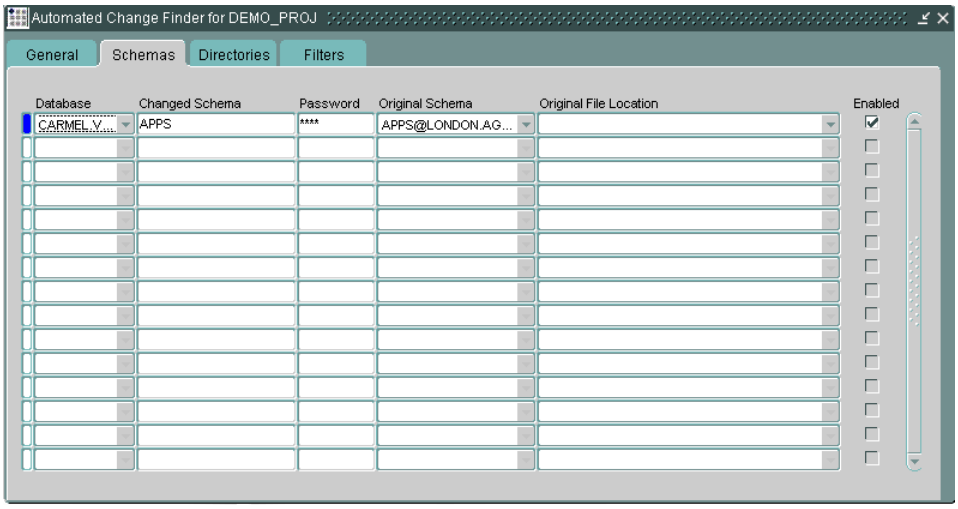
Run after	This command is used to run a program after Automated Change Finder runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).
Summary Report (button)	Displays a report of all differences found.
View Changes (button)	Displays the View Changes window (p. 75).
Schedule (button)	Begins the process of scheduling Automated Change Finder. Displays the Schedule window (p. 142).

View Changes



Lists all files that were tested; files marked **Identical** have not changed.

Schemas



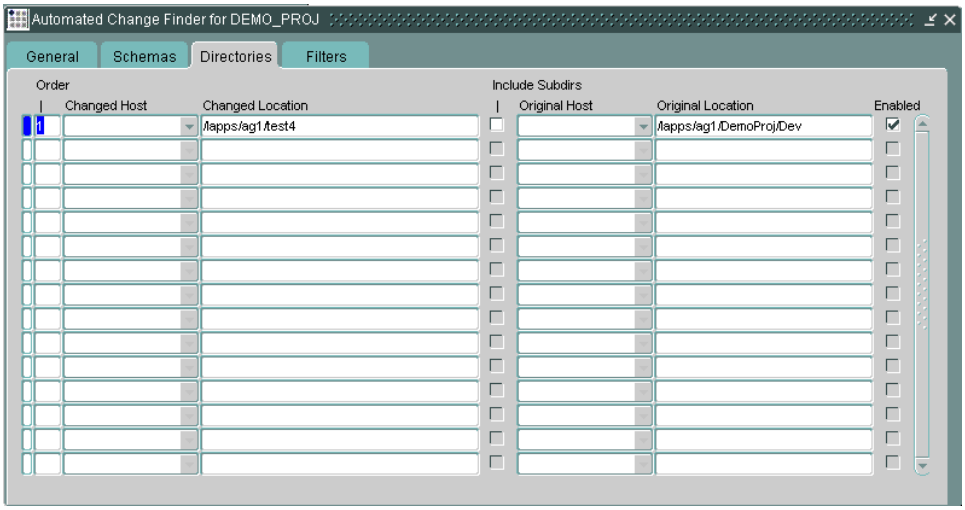
This tab lets you specify the schemas that should be monitored for changes.

> The `INIT.ORA` file for the database where Codebase resides must have `global_names` set to `FALSE`.

The tab contains these fields:

Database	Database that holds the **Changed Schema**.
Changed Schema	Schema to monitor.
Password	**Changed Schema**'s password.
Original Schema	Schema that **Changed Schema** will be compared to.
Original File Location	Full path to the directory that holds the SQL scripts and packages related to **Original Schema**.
Enabled	When checked, this schema will be monitored.

Directories



This tab lets you specify the directories that should be searched for files with extensions specified in the Automated Generation window's Extensions tab (p. 111). All values entered in this tab will take effect immediately, without requiring a restart of the Automated Change Finder.

Be sure to include any directory that may contain referenced forms or attached libraries. An error will occur whenever a referenced form or attached library cannot be found in the directories you've specified.

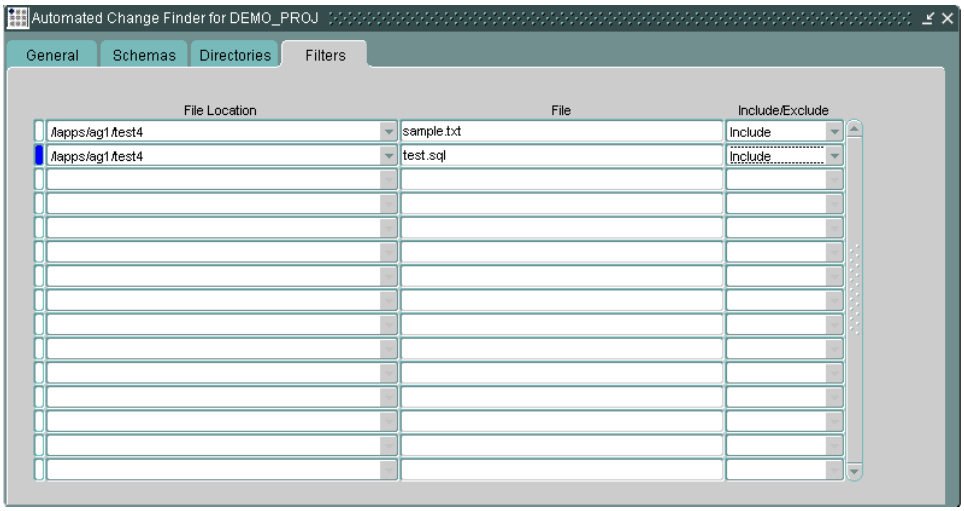
Also, be sure to specify the file locations for all Oracle Modules that you have linked your Custom Module against. One commonly overlooked directory is the `FND` directory that contains the `TEMPLATE.fmb` file.

If a referenced form is modified, and is being used in another form, the latter will automatically be compared to its previous version. Thus, the modification to the referenced form will appear in the difference report (assuming the modified object was being referenced).

Order	Automated Change Finder will process the locations in this order. The order becomes important when a duplicate file is found in multiple directories. The first occurrence of a duplicate file is processed and all other occurrences are skipped.
Changed Host	Host used to access **Changed Location**.

Changed Location	Enter valid directories that contain the form, library, report, and text files. You may enter the directories with or without a "\" at the end. These same file locations should exist in the `FORMS60_PATH` environment variable. You cannot query records in this tab. The entire record set will always be displayed. To refresh this information, you can select **Edit > Clear > Form** from the menu bar. If the directory is not readable, you will receive an error.
Include Subdirs	Check this box to include all subdirectories of the location. If the same file exists in more than one subdirectory, the first occurrence of the file will be used.
Original Host	Host used to access **Original Location**.
Original Location	Directory that holds files to be compared to those in **Changed Location**.
Enabled	When checked, the files in **Changed Location** are processed.

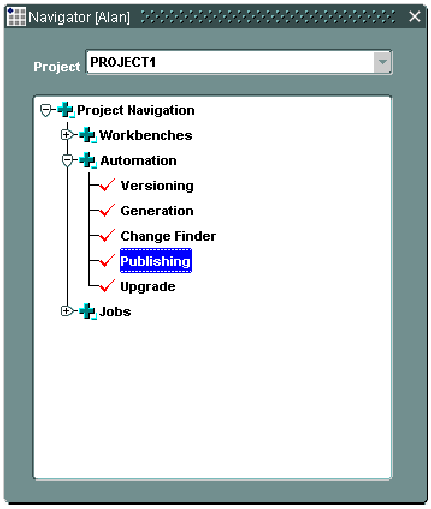
Filters



To force the processing of specific files, select their **File Locations**, enter their names in the **File** fields, and set Include/Exclude as desired.

The list of values of file locations will be populated from the Directories tab, Changed Location values.

Automated Publishing



Automated Publishing produces documentation of your forms, reports, menus, libraries, and SQL code. You can use these reports for code reviews, exception reporting, documentation, disaster recovery and impact analysis. They are also used by Object Finder's Subclassed Forms (p. 86) and Attached Libraries (p. 88) tabs.

Automated Publishing processes files with the following extensions:

`FMB, MMB, OLB, RDF, PLL`

Any extension specified in the Automated Generation window's Extensions tab (p. 111) with the **Text File** checkbox selected

If you want to use Automated Publishing to monitor several Environments, such as development, test and production, create a separate project for each.

When publishing a form, Automated Publishing does not publish the libraries attached to the form.

If a referenced form is modified and is being used in another form, the latter form will automatically be re-published to show changes made on the referenced object(s). Thus, the modification to the referenced form will appear in the Form Document report (assuming the modified object was being referenced).

If you set up Automated Publishing to process only Automated Change Finder's Last Version Location directory, run Automated Change Finder before running Automated Publishing.

To use Automated Publishing:

As with all automated processes, you must have the Projects: Users tab's (p. 27) **Admin** privilege to configure and schedule this process, and the tab's **Jobs** privilege to view its execution status and logs.

1. In the Navigator window (p. 10), select your **Project**.
2. Double-click **Project Navigation > Configurations > Automated Publishing**.
3. Configure the window's tabs (General (p. 127), File Locations (p. 131), Properties (p. 133) and Filters (p. 134)).
4. Click the General tab's (p. 127) **Schedule** button. The Schedule window (p. 142) appears.
5. Set all scheduling options as desired; you can run Automated Publishing once, or repeatedly. For example, you can publish documentation nightly to provide developers with up-to-date information.

   Never run more than one instance of Automated Publishing at the same time, unless the instances are for separate projects. Automated Publishing can run concurrently with any other process, except Change Finder.

6. Click the **Schedule** button. A new job is created; its activities can be monitored in the View Inside Jobs window (p. 136) and its log (p. 140).

   During its initial run, Automated Publishing creates a report for every source file it processes. This process may be time-consuming, depending upon the project's size. Most subsequent runs require significantly less time, because Automated Publishing processes only the files that have changed since its last run.

   When errors occur, they will be written to the log (p. 140). The program can be restarted at anytime after a problem or system crash has occurred. The program will start processing where it left off.

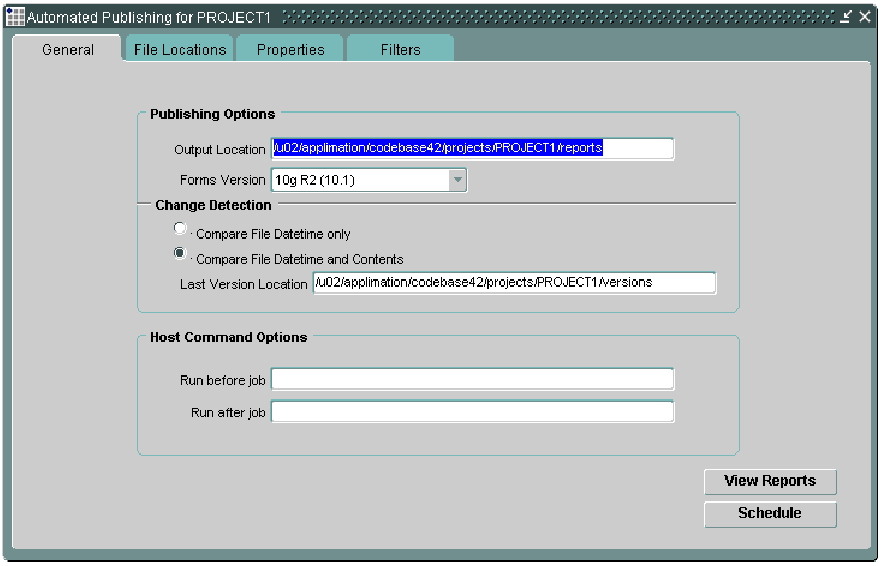
7. After verifying that Automated Publishing has run successfully, view the results by clicking the General tab's View Reports button (p. 130).

File Date depends on whether Automated Change Finder has processed the file. If it has, File Date is the file's modification date at the time Automated Change Finder processed it. If it has not, File Date is the file's modification date at the time Automated Publishing processed it.

Also, you can now use Object Finder's Subclassed Forms (p. 86) and Attached Libraries (p. 88) tabs.

8 If you have scheduled repeated execution of Automated Publishing, check the log (p. 140) periodically to ensure that no errors occur.

General



This tab lets you configure Automated Publishing (p. 124).

If a form is modified and its objects are being subclassed in another form, the latter form will automatically be republished to show changes made to the subclassed object(s). Thus, the modification to the subclassed objects will appear in the documentation generated by Automated Publishing.

> The values entered in this tab will not take effect until Automated Publishing is restarted.

This tab contains the following fields and buttons:

Publishing Options

Output Location	Enter a directory where the reports should be stored. All developers must have access to this directory so that they can view the reports via the **View Reports** screen. The directory must exist and be readable and writeable, otherwise you will receive an error message.
Forms Version	The environment's version of Oracle Forms.

Compare File Datetime Only

Select this radio button when you want files to be considered changed based solely on the system file date. The system file date will be compared to the file date stored internally by Automated Publishing. If the file dates are different, then Automated Publishing will consider the file to be changed. This option will optimize the amount of disk space that is required to run the Automated Publishing.

> There are cases when a file's date will change but its contents remain the same. One such scenario is when a file gets checked out of a traditional version control system with a new timestamp. Another scenario occurs on some Microsoft Windows platforms when the **Automatically Adjust for Daylight Savings** option is turned on causing all files being viewed through the Windows software to change by one hour. If either of these scenarios occur, several modules may be analyzed needlessly. This problem could cause extensive processing to occur. These scenarios are correctable, but if you find a scenario in your environments that cannot be easily corrected, use the **Compare File Datetime and Contents** option.

Compare File Datetime and Contents

Select this radio button when you want content to be considered changed based on both the system file date *and* the file's contents. The system file date will be compared to the file date stored internally by Automated Publishing. If the file dates are different, the contents of the file will be compared to a version that was previously saved by Automated Publishing (see **Last Version Location**). When the file date *and* contents both change, the module will be re-published. This option requires more disk space, to store an additional version of each file. This option will optimize the performance of Automated Publishing.

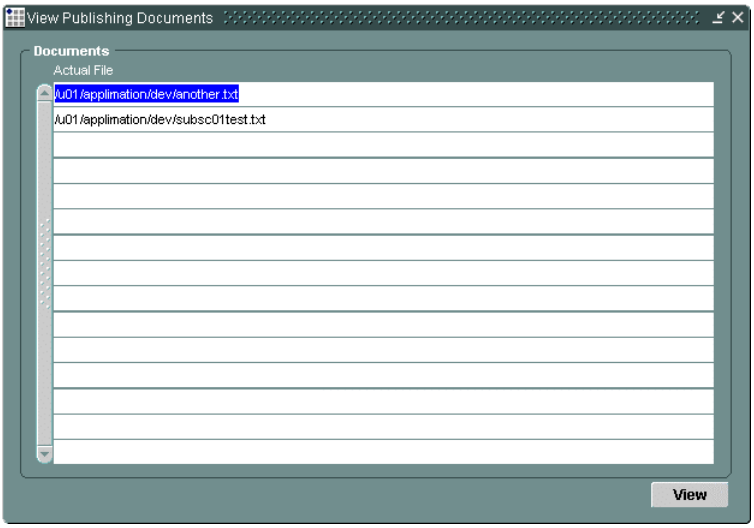
Last Version Location	This directory will be defaulted to a directory created by Codebase on your Codebase Server at the time the project was created. This directory stores the last version of each module. Make sure there is plenty of disk space for one version of each file. This directory must be a different location than the ones specified in other Codebase windows. If you decide to change the value, the directory must exist, be readable and writeable. Otherwise you will receive an error message.

Host Command Options

Run before	This command is used to run a program before Automated Publishing runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).
Run after	This command is used to run a program after Automated Publishing runs. This command is useful for interfacing to other programs and will wait until the called program finishes. Any valid operating system command can be entered here. The `sub_the_user_login` runtime variable is available to obtain the login username, password, and database connection string. The `sub_env_passwd_???$` runtime variable is available for obtaining the password to a **Versioning Schema** in Automated Generation (see Schemas (p. 107) for more information).

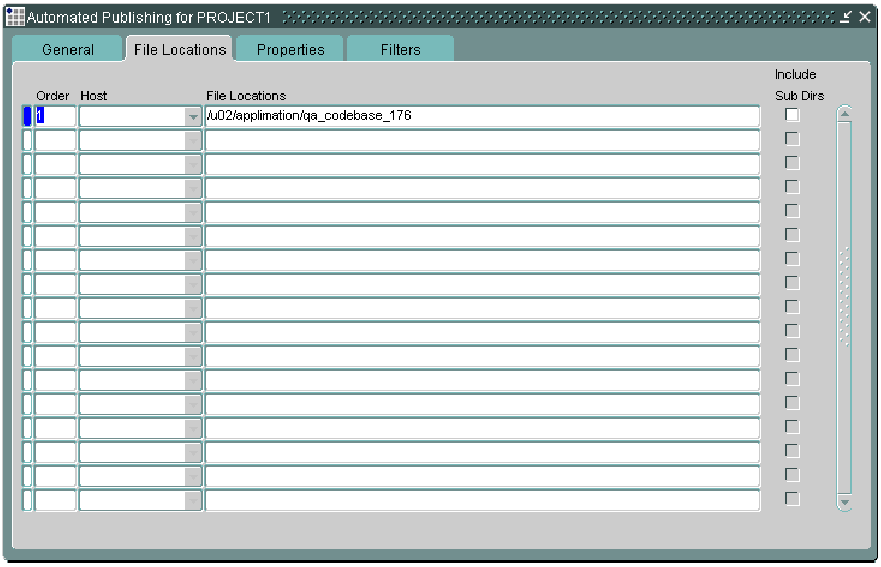
View Reports (button)	Displays the View Reports window (p. 130).
Schedule (button)	Begins the process of scheduling Automated Publishing. Displays the Schedule window (p. 142).

View Reports



This window lists all modules (i.e., files and database objects) documented by Automated Publishing (in the currently selected project). To view a document, click the **Actual File**, and then click **View**.

File Locations



This tab allows you to specify the directories that should be searched for files with an extension of:

`FMB`, `PLL`, `RDF`

Any extension defined in the Automated Generation window's Extensions tab (p. 111) that has **Text File** checked.

All values entered in this tab will take effect immediately (restarting Automated Publishing is not necessary). To refresh the information in this tab, select **Edit > Clear Form** from the menu bar.

Automated Publishing will save all form and library dependency information for the `FMB` and `PLL` files contained within the directories in this tab. Be sure to include any directory containing files that you want to view via the **Object Finder** screen.

Unlike the Automated Change Finder, Automated Publishing does not look at the links you have created in the Link Modules window (p. 51). Thus, each and every form, report, library, and text file that is found in the directories you specify will be processed (including the Oracle vanilla files). This is different than the Automated Change Finder, since the Automated Change Finder only processes linked files when **Processing Options** is set to `Process Only Linked Modules`.

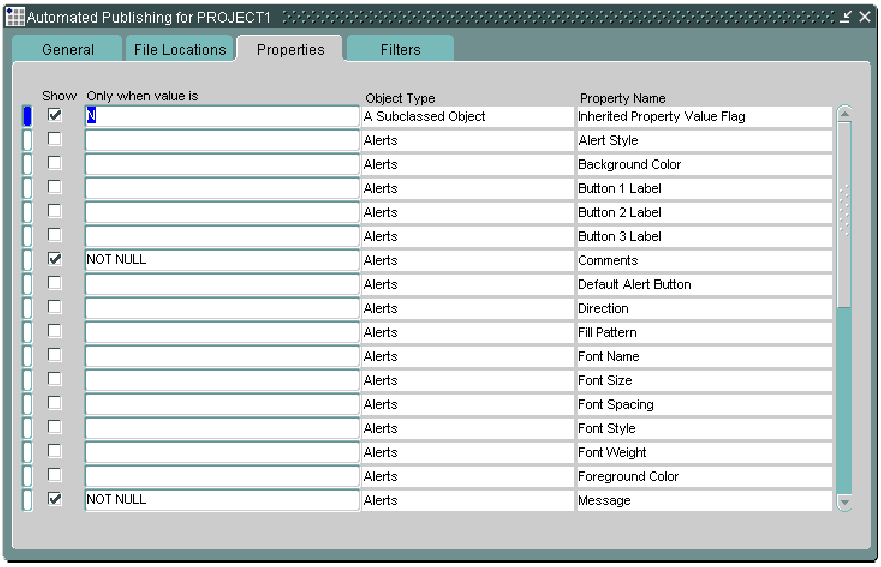
When entering directories, be sure to include any directory that may contain a subclassed form. Otherwise, an error will occur when subclassed forms cannot be found in the directories you specified. When publishing a form, libraries attached to the form will not be published along with the form because the

Oracle E-Business Suite users: You may need to specify Oracle's standard forms' directories, so subclassed forms can be found for your custom forms. If you do not want to publish all of Oracle's standard forms, use the **Exclusions** tab to exclude them.

The tab contains these fields:

Order	Enter a sequence number for the **File Locations**. This sequence will determine the order in which the locations are searched. The order becomes important when a duplicate file is found in multiple locations; the first occurrence is processed and all others are skipped.
Host	Unix/Linux users: Select the Host used to access **File Locations**.
File Locations	Enter valid directories that contain the form, library, report, and text files. You may enter the directories with or without a "\" at the end. These same file locations should exist in the `FORMS60_PATH` environment variable or some Oracle tools will not work properly (see Oracle installation manual) when searching for referenced forms and attached libraries. The directory must be readable or you will receive an error.
Include Subdirs	Check this box to include all subdirectories for the entered file location. This will eliminate ongoing maintenance when directories get added under the parent directory. If the same file exists in more than one subdirectory, the first occurrence of the file will be used.

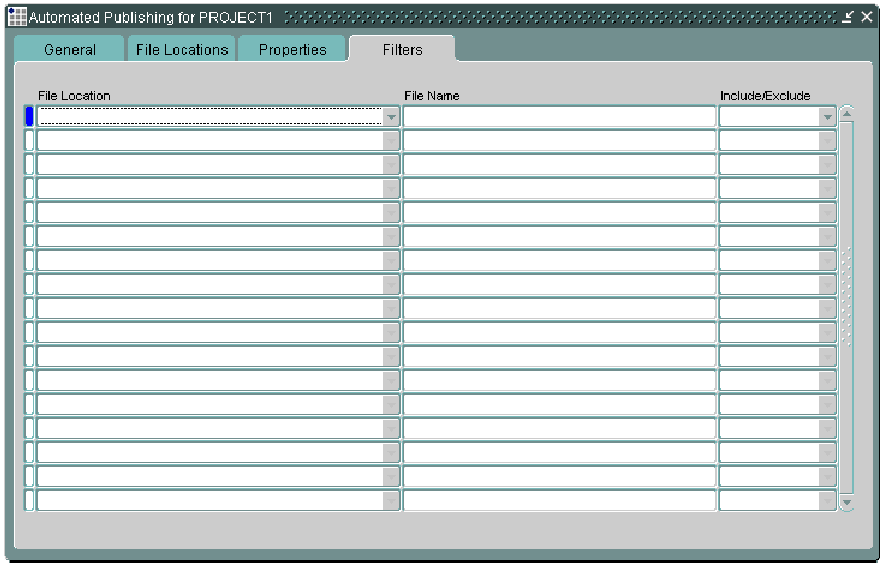
Properties



This tab lets you specify the properties that will appear in the documents created by Automated Publishing. These settings apply to Forms, Reports, Object Libraries, and PL/SQL Libraries. The settings will also be used when manually creating a document using the Change Finder. A document will also be created by comparing a file and directory location to itself in the Change Finder; likewise for reports, libraries, and text files.

Show	Check this box to document the property.
Only when value is...	Adding a value in this field will further restrict the occasions when the property is documented. Entering `NULL` will document the property only when it has no value. Entering `NOT NULL` will document the property when it does have a value. Specific values can also be entered, such as `True`, `False`, `1`, `0`, etc. When `!` is the first character, it is a NOT operator (e.g., `!0` means "not a value of zero," `!False` means "not a value of False," etc.). Use `%` as a wildcard. This field has no effect for the **Object Type** `A Subclassed Object`.
Object Type	Displays the type of Forms/Reports Designer object that the property belongs to (e.g., `Data Blocks`, `Triggers`, `Summary Columns`, `Frames`, etc).
Property Name	Displays the **Property Name** that appears in the Forms/Reports Designer.

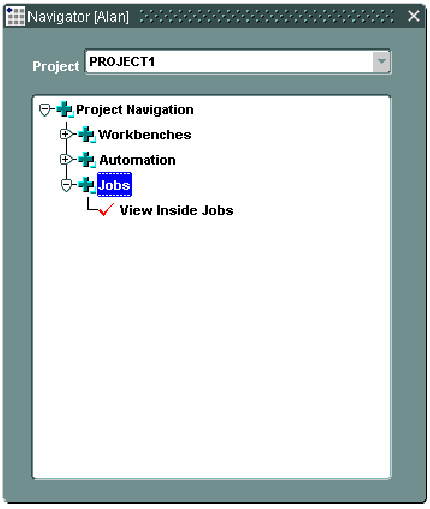
Filters



To force the processing of specific files, select their **File Location**s, enter their names in the **File** fields, and set Include/Exclude as desired. You may use asterisks (*) as wildcards in the **File** field.

6

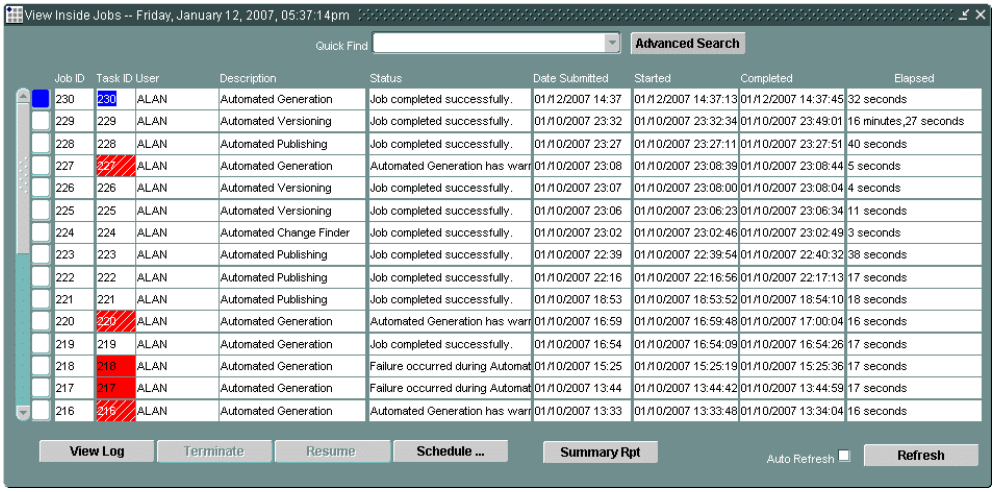
Jobs



Many Codebase actions take place in the background, leaving you free to move on to other tasks. These background actions are called **Jobs**.

In This Section

View Inside Jobs



Integra Codebase schedule jobs to perform most of its actions. This window lets you monitor, control and schedule jobs.

To display the window, double-click **Jobs > View Inside Jobs** in the Navigator window (p. 10).

The **Quick Find** dropdown lets you display:

All My…	All…
Jobs	Jobs
Jobs Submitted Today	Pending Jobs
Pending Jobs	Running Jobs
Running Jobs	Scheduled Jobs
Scheduled Jobs	Cancelled Jobs
Cancelled Jobs	Failed Jobs
Failed Jobs	Warning Jobs
Warning Jobs	

The **Advanced Search** button lets you search for:

All Jobs **My Jobs** **Jobs with Specific Owner**	View jobs created by all users, just yourself, or another user.

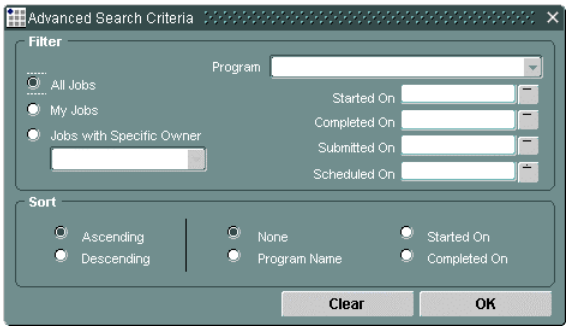
Program	View jobs containing the specified program.
Started On	View jobs begun on the specified date.
Completed On	View jobs finished on the specified date.
Submitted On	View jobs submitted on the specified date.
Scheduled On	View jobs that are scheduled to start on the specified date.
Sort: **Ascending** **Descending** **None** **Program Name** **Started On** **Completed On**	Sort jobs in the specified order, on the specified column.

The window also contains these fields and buttons:

Quick Find	*(Described above)*
Advanced Search (button)	Displays the Advanced Search window (p. 138).
Job ID	Each job is assigned a number when it is submitted.
Task ID	Each job is also assigned a task number when submitted.
User	The Codebase user who submitted the job.
Description	The first program to be executed.
Status	The job's current execution status. The **Warning** status indicates that important messages can be found in the job's log (p. 140). To view the log, click the **View Log** button.
Date Submitted	The date and time when the job was submitted.
Started	The date and time when job execution began.

Completed	The date and time when job execution ended.
Elapsed	The difference between **Completed** and **Started**.
View Log (button)	Displays the Log window (p. 140).
Terminate (button)	Ends execution of the selected job as soon as possible.
Resume (button)	Continues the execution of a paused job.
Schedule... (button)	Displays the Schedule window (p. 142).
Summary Rpt (button)	Displays a report that summarizes the activity in the selected job.
Auto Refresh	When checked, the job data in this window is refreshed periodically.
Refresh (button)	When clicked, the job data in this window is refreshed.

Advanced Search



The window contains these fields and buttons:

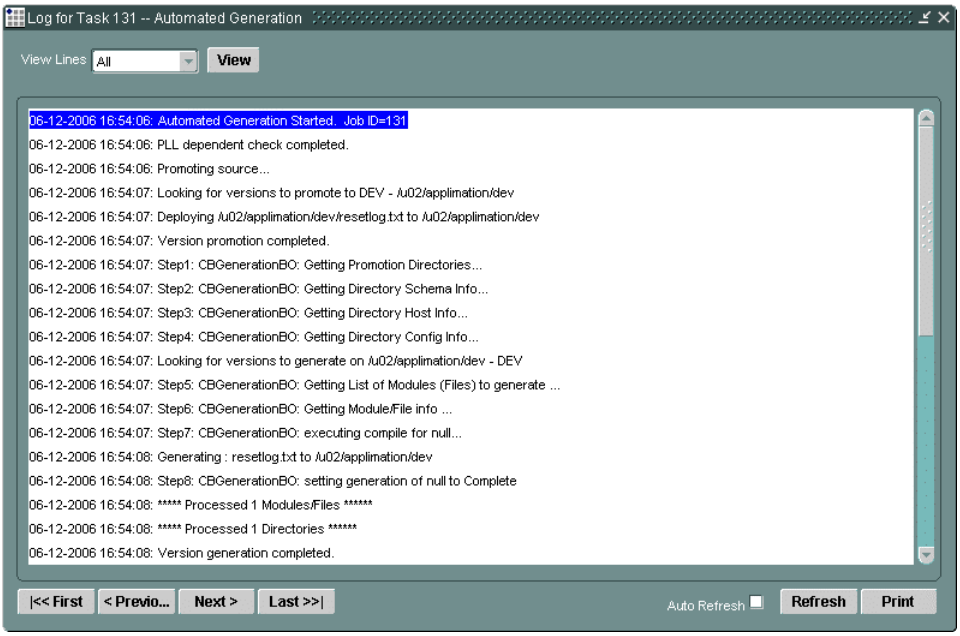
Filter

All Jobs **My Jobs** **Jobs with Specific Owner**	Select whose jobs you want to find.
Program	Find jobs containing the selected program.
Started On	Find jobs started on this date.
Completed On	Find jobs completed on this date.
Submitted On	Find jobs submitted on this date.
Scheduled On	Find jobs scheduled to begin on this date.

Sort

Ascending **Descending**	Choose sort direction.
None **Program Name** **Started On** **Completed On**	Choose the field to sort on.
Clear (button)	Resets the window's fields to their default values.
OK (button)	Performs the search.

Log



The View Inside Jobs window's (p. 136) **View Logs** button displays this window, which contains the execution log of the selected job.

Errors are listed in the log. They typically appear when:

- `FORMS60_PATH` (or `FORMS90_PATH`) does not contain the proper directory locations for referenced files
- Files are corrupted

Be sure to periodically check the log to ensure that no errors have occurred.

The window contains these fields and buttons:

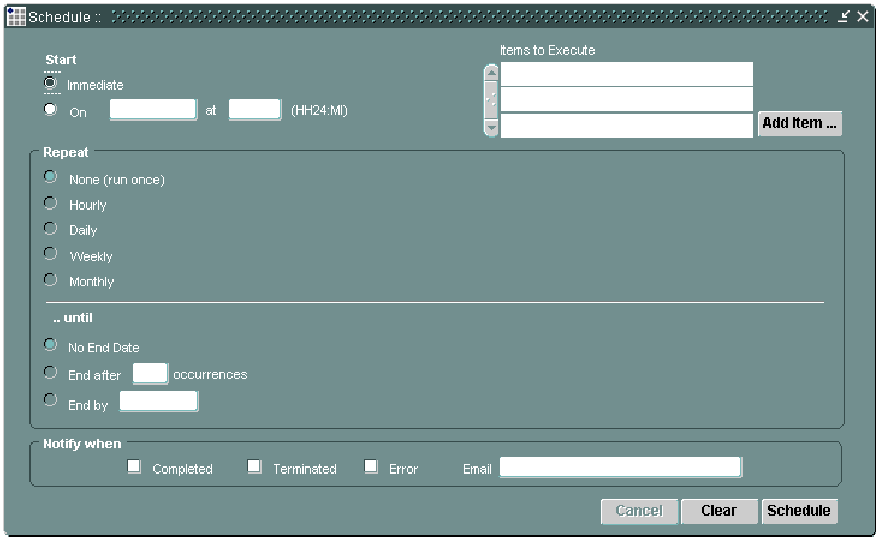
View Lines

View (button)

To filter the lines listed, change the **View Lines** dropdown, fill in any fields that appear, and click **View**.

First (button)	Displays and selects the log's first row.
Previous (button)	Displays and select the row before the one that is currently selected.
Next (button)	Displays and select the row after the one that is currently selected.
Last (button)	Displays and selects the log's last row.
Auto Refresh	When checked, the log is refreshed periodically. Useful when monitoring the progress of a job.
Refresh (button)	Refreshes the log. Useful when monitoring the progress of a job.
Print (button)	Displays a printer-friendly version of the log.

Schedule



The window contains these fields and buttons:

Start

Immediate On	Choose to start the job immediately or on a certain date and time.

Items to Execute

Items to Execute	List of items that will be executed.
Add item... (button)	Displays the Add Item window (p. 144).

Repeat

None (run once)	Specify how often job should be re-run.
Hourly	
Daily	
Weekly	
Monthly	

...until

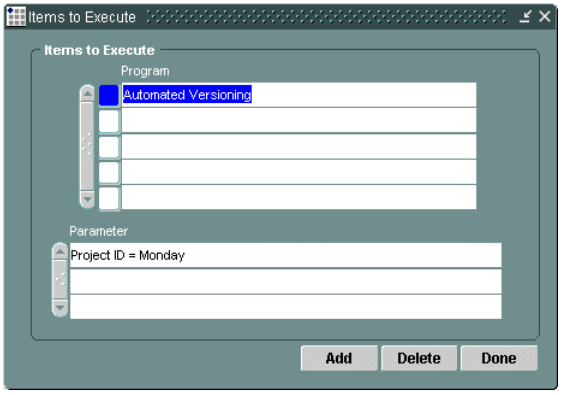
No End Date	If job will repeat, specify when the repetition will end.
End After _ Occurrences	
End By	

Notify when

Complete	Sends a notification to **Email** when the job has completed successfully.
Terminated	Sends a notification to **Email** when the job has been terminated by a user.
Error	Sends a notification to **Email** when the job has ended in error.
Email	Recipients of **Completed**, **Terminated** and **Error** notifications. Separate multiple addresses with commas (,).

Clear (button)	Resets all fields to their default values.
Schedule (button)	Submits the job.

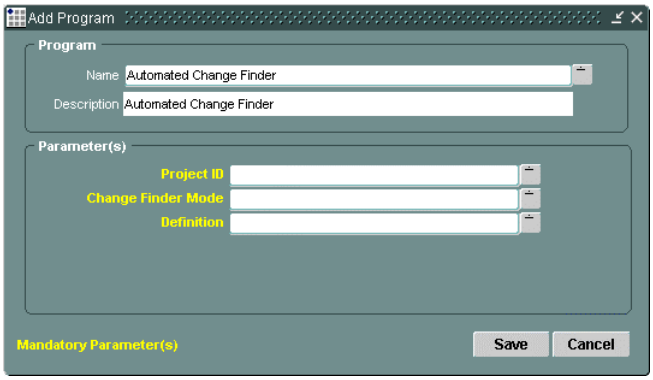
Add Item



This window contains these fields and buttons:

Program	List of items that will be executed.
Parameter	List of execution parameters for the selected **Program**.
Add (button)	Displays the Add Program window (p. 144).
Delete	Removes the selected **Program** from the list.
Done	Saves your changes and returns you to the Schedule window (p. 142).

Add Program



This window contains these fields and buttons:

Program

Name	Item that will be executed.
Description	Description of item that will be executed.

Parameters

The fields in this section vary, depending on the **Program**.

Save (button)	Saves your changes and returns you to the Add Item window (p. 144).
Cancel (button)	Discards your changes and returns you to the Add Item window (p. 144).

Change Passwords

Individual Codebase users' passwords can be changed by the users themselves or by administrators, either through the Codebase user interface or a "backend" script. The Codebase schema password can also be changed through the use of a script, and the Codebase Agent password can be changed via the Codebase application.

To make any of these changes, first complete these steps:

1. At the Linux prompt, or through the use of scp, open the *CODEBASE_HOME*/Tools directory.
2. Use SQLPlus to log in to the Codebase schema (Cb_admon/db).
3. Run the package specification and package body script (cb_chgpwd_pkg.pks and cb_chgpws_pkg.pkb).

The user who is logged on to Codebase can change his own password:

1. Having logged on to Codebase 4.2.2, a user clicks on Action > Change Password.
2. The user then enters his old and new password, and saves his work.

For an administrator to change a user password:

1. The administrator logs on to Codebase 4.2.2 and, in the Administration project, opens the Configuration > General form.
2. In the Users tab, the administrator changes the desired user's password and saves the work.

To change a user's password through use of a script, enter the Codebase schema and run the following command, in which *username* identifies the user and *password* is the new password:

```
BEGIN
cb_chgpwd_pkg.chg_user_password ('username','password');
END
```

To change the Codebase schema password, enter the Codebase schema and run the following script:

```
BEGIN
cb_chgpwd_pkg.chg_schema_password ('schemaName','password');
END
```

To change the Codebase schema password, enter the Codebase schema and run the following script:

1. Log on to Codebase 4.2.2 and open the Administration > Configuration > General > Database tab.
2. Selecte the database for which the Codebase agent password is to be changed.
3. On the right side of the screen, select the Recreate check box, change the password, and provide the System schema password for the current database.
4. Save the Form.

7

Help

Selecting **Help** from the menu bar displays the following options:

Help	Displays a list of common keyboard shortcuts.
Display Error	Displays a list of recent error messages.
View Inside Jobs	Displays the View Inside Jobs window (p. 136).
Show Navigator	Displays the Navigator window (p. 10).
Window Help	Displays help about the window currently in the foreground. Equivalent to clicking  .
About...	Displays copyright information and version numbers.