HPCC Systems®

The ECL IDE and HPCC Client Tools

Boca Raton Documentation Team

LexisNexis® Risk Solutions

The ECL IDE and HPCC Client Tools

Boca Raton Documentation Team

We welcome your comments and feedback about this document via email to <docfeedback@hpccsystems.com>

Please include **Documentation Feedback** in the subject line and reference the document name, page numbers, and current Version Number in the text of the message.

2015 Version 5.4.2-1

Overview

This manual contains documentation for the set of Client Tools for use with the LexisNexis HPCC. These tools include:

ECL IDE	An integrated development environment for ECL programmers to create, edit, and execute ECL code.
ECL Debugger	A development tool integrated into the ECL IDE, which is used to debug queries as they are developed.
ECLPlus	Command line ECL execution tool to facilitate automation of ECL Code execution.
ECL	Command line ECL tool.
ECL Compiler	Command line ECL Compiler
DFUPlus	Command line Distributed File Utility management tool, facilitate automation of data file spray, despray, and other common file handling tasks.
ESDL	Command line ESDL management tool.

Documentation Conventions

ECL Language

Although ECL is not case-sensitive, ECL reserved keywords and built-in functions in this document are always shown in ALL CAPS to make them stand out for easy identification.

Example Code

All example code in this document appears in the following font:

```
MyECLFileName := COUNT(Person);
// MyECLFileName is a user-defined ECL file
// COUNT is a built-in ECL function
// Person is the name of a dataset
```

ECL file names and record set names are always shown in example code as mixed-case. Run-on words may be used to explicitly identify purpose in examples.

Actions

In step-by-step sections, there will be explicit actions to perform. These are all shown with a bullet or a numbered step to differentiate action steps from explanatory text, as shown here:

- Keyboard and mouse actions are shown in small caps, such as: DOUBLE-CLICK, or press the ENTER key. word.
- On-screen items to select are shown in boldface, such as: press the **OK** button.

Installation

The installation program installs all client tools, including the ECL IDE, ECLPlus, DFUPlus, and the ECL Command line tools.

1. From the HPCC Systems® download page, http://hpccsystems.com/download/free-community-edition/client-tools

   Download the appropriate Client Tools for your Operating System. (available for CentOS, Ubuntu, Mac OSX, or Windows)

2. Install the client tools software to your machine.

   Windows:

   Run the executable file, for example: hpccsystems-clienttools_community-4.X.X-XWindows-i386.exe on your machine. Follow the prompts to complete the installation.

   RPM-Based Systems (CentOS/RedHat):

   An RPM installation package is provided. Install RPM with the -Uvh switch, the U or upgrade will perform an upgrade if a previous version is already installed.

   ```
   sudo rpm -Uvh <rpm file name>
   ```

   Debian-Based Systems (Ubuntu):

   For Ubuntu installations a Debian package is provided. To install the package, use:

   ```
   sudo dpkg -i <deb filename>
   ```

   Mac OSX:

   Run the installation file, for example: hpccsystems-clienttools_community-4.X.X-XDarwin-x86_64.dmg. Follow the prompts to complete the installation.

The ECL IDE

Introduction

ECL IDE is the simple and easy way to create Queries into your data, and ECL files with which to build your queries. You build your Queries and files (see the *ECL Language Reference* and the *ECL Programmer's Guide)* using Enterprise Control Language (ECL). ECL has been designed specifically for use in huge projects where many things can be similar.

ECL's extreme scalability comes from a design that allows you to leverage every query you create for re-use in subsequent queries as needed. To do this, ECL takes a *dictionary* approach to building queries wherein each ECL definition defines an ECL file. Each previously defined ECL file can then be used in succeeding ECL File definitions—*the language extends itself as you use it*.

ECL IDE is an ECL programmer's tool. Its main use is to create the ECL files and is designed to make ECL coding as easy as possible.

ECL files

These are the basic building blocks from which you create queries into your data. ECL files are extensively defined in the *ECL Language Reference.*

You create an ECL file by coding an expression that defines how a calculation or record set derivation is to be done. Once an ECL file is defined you may use it in succeeding ECL file definitions, making each succeeding ECL file more and more highly leveraged upon the work you have done before. This results in extremely efficient query construction.

Built-in Functions

ECL IDE also has all the ECL built-in functions available to you for simple point-and-click use in your query construction. For example, the Standard String Library (Std.Str) contains common functions to operate on STRING fields such as the **ToUpperCase** function which converts characters in a string to uppercase.

These are installed as plug-ins and are in the ecllib folder shown in the Repository window.

How are all these used to build Queries?

ECL IDE allows you to mix-and-match your data with any of the ECL built-in functions and/or ECL files you have defined to create Queries. Because ECL files build upon each other, the resulting queries can be as complex as needed to obtain the result.

Once the Query is built, you send it to a High Performance Computing Cluster (HPCC) which processes the query and returns the result—extremely quickly.

Complex Queries that may have taken weeks to format, program, and execute using old-style mainframe data-mining tools can literally execute and return the result in seconds.

Configuration Files

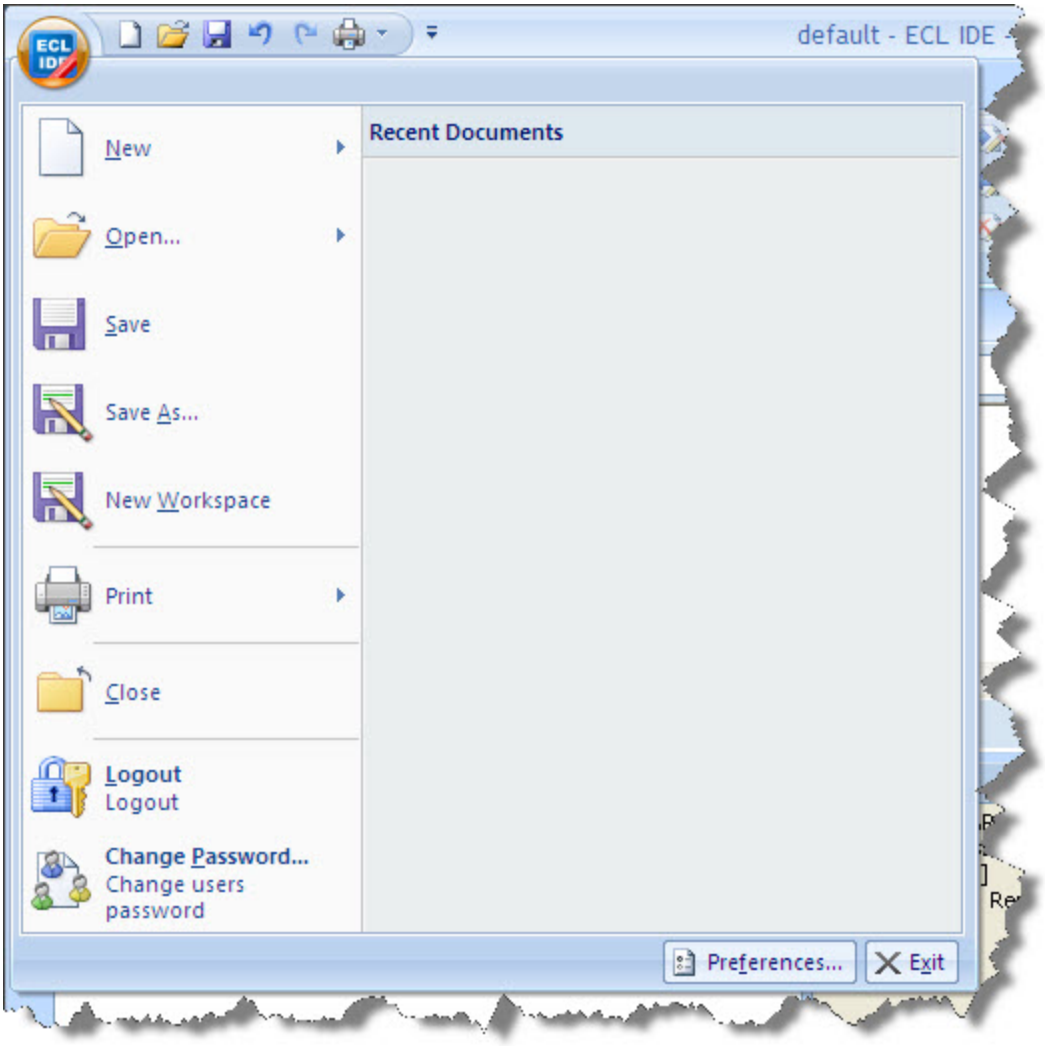
ECL IDE uses configuration files (.CFG) to store the information for environments to which it will connect. Create multiple configuration files to connect to different environments. Use the **Save As** option on the **Preferences** window to create a new configuration file.

To create a configuration, you need to know the following:

- The IP address of the ESP Server.
- If services are not using the default ports, you will also need to know the port bindings for the following ESP Web services:
  - WsTopology (default port is 8010)
  - WsWorkunits (default port is 8010)
  - WsAttributes (default port is 8145) **Note:** Not part of VM or Community Edition
  - WsAccount (default port is 8010)
  - WsSMC (default port is 8010)
  - WsFileSpray(default port is 8010)
  - WsDfu(default port is 8010)

You can find the ESP Server's IP address and the bindings for service ports using the System Servers section in ECL Watch.

To create or edit configuration files:



Press the **Preferences** button on the Orb button drop menu.

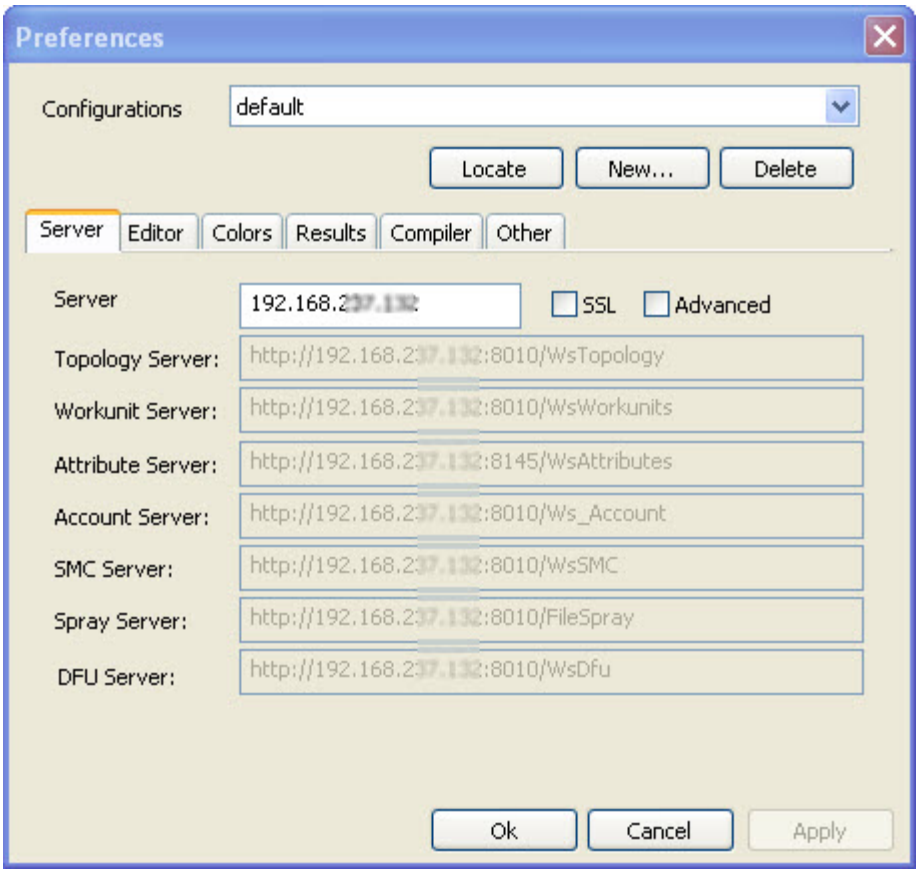
or

- Press the **Preferences** button on the Login window.

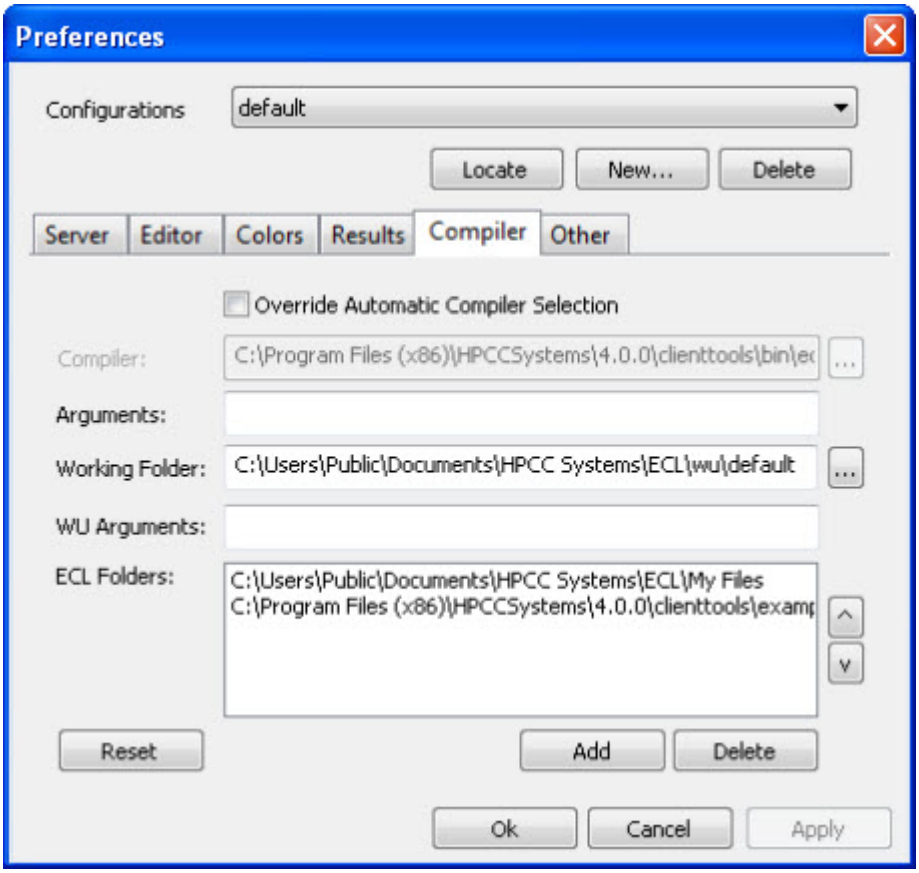
  Note: Some options, such as the Server settings, can only be changed by accessing the preferences through the Login window.

The Preferences window displays:

- Select the **Server** tab and enter the IP address of the ESP in the Server field, but do not include the port details:



- Select the **Compiler** tab. The compiler details are automatically specified:



- Specify your Working Folder.

  When you are running your locally, your queries are compiled and stored in this location.

  To compile a query locally rather than on the thor or hthor of your environment, select **Local** as the **Target** before pressing **Submit** on the **Builder** window.

  If you are running under Windows and want to compile locally, install the Microsoft VS 2008 C++ compiler (either Express or Professional edition) and Linux users need GCC.

- Specify your ECL Folders.

  When creating queries in ECL IDE, you must save the ECL file before you can run the query. Specify the locations on your computer where you want the ECL files to be saved. All folders specified here are listed in the **Repository** window in ECL IDE where you can get easy access to all your files.

  Sample ECL files which are supplied with ECL IDE are automatically installed onto your computer when you download ECL IDE. They are downloaded to the My Documents folder on your computer and the location of this Working Folder is automatically added into the configuration.

- If you are creating a new configuration, click **New...** and enter a name without spaces.

- If you are editing an existing configuration, click **Apply** and then **OK** to save your changes.

  When you have completed your configuration changes, you are prompted to relogin.

Using the Preferences Window

Server Tab

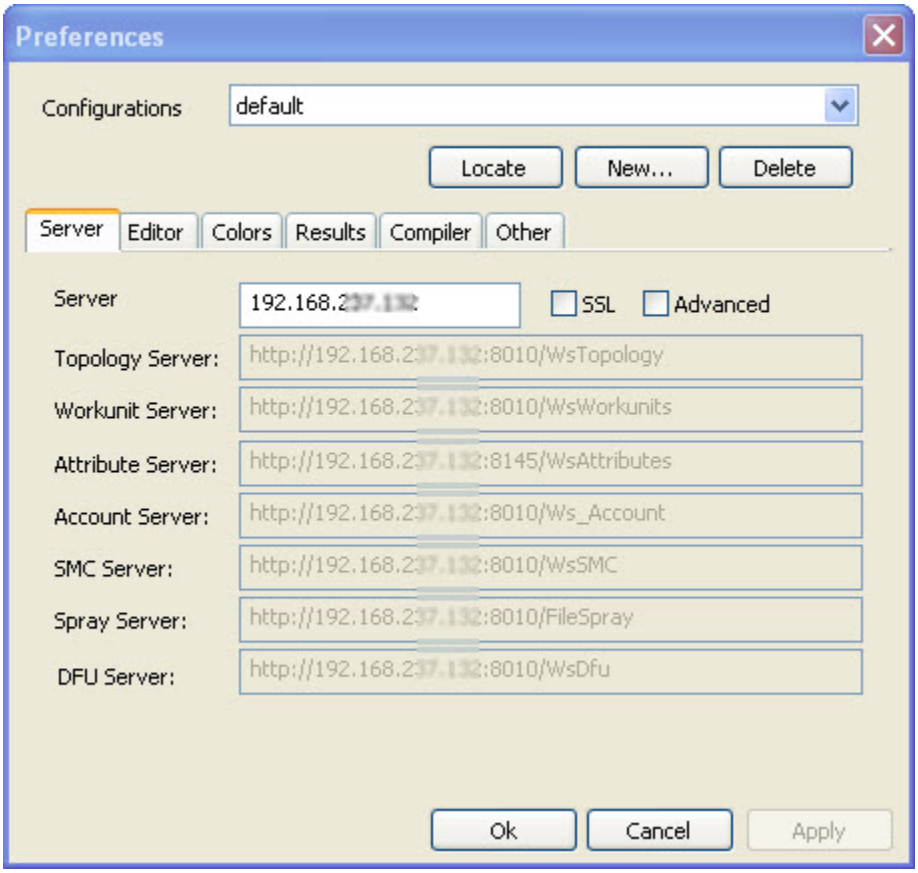
The **Server tab** has two modes for setting the server information: Normal and Advanced. Use the normal mode when all ESP web services are deployed using default settings.

In normal mode, you need only supply the ESP server's IP address or DNS machine name. The remaining values are automatically completed.

If you are creating a connection using Secure Socket Layer (SSL) to connect, check the **SSL** box. See the *Using SSL* section in this manual for more information.

Use Advanced Mode to edit individual values for the ESP Server connections.

Check the **Advanced** box to use Advanced Mode.



The Server settings, can only be changed by accessing the preferences through the Login window.

Press **Apply** and then **OK** to save the current information to a new configuration file. All available configuration files are shown on the **Configurations** drop list when you log in and you simply select the one you want ECL IDE to use to connect to an HPCC environment.

	Press the **Locate** button to find the folder containing a configuration file which you have previously created and saved onto your computer.

Editor tab

The **Editor tab** allows you to configure the editor to your preference.

Tab Width (chars)	Specify the number of character spaces to use for a tab. The default is 2.
Auto Save (secs)	Specify automatic save interval in seconds. The default is 10.
Maintain Indent	Check this box to maintain the level of indention when creating a new line in the editor.
Line Numbers	Check this box to display line numbers in the editor.
Tree	Check this box to display code structures in a collapsible tree at the side of the editor.
Open MDI children Maximized	Check this box if you prefer child windows to open maximized within the application frame.
Tooltips	Check this box to enable tooltip display.
Font	Specify the font to use in the editor. The default is Courier New.
Size	Specify the font size to use in the editor. The default is 10.
Keep Repository Synchronised	Check this box if you want to ensure that when an ECL file window is selected, the ECL file is auto selected in the repository.
Auto Complete on Period	Check this box to show a list of available ECL files whenever an existing folder name is typed.
Double Click Selects Qualified Label	Check this box to ensure that on a double click, the entire folder and ECL file name is selected. When unchecked, only the folder name is selected on a double click.
Tip:	You should use a fixed-width font to maintain alignment in ECL source code.

Colors tab

The **Colors tab** allows you to configure the text, foreground and background colors to your preference.

Element	Specify the Element the formatting should be applied to. The drop list shows all available elements.
Defaults	Press this button to reset all values to the default color settings.
Font	Specify the font to use in the Results window. The default is Arial.
Size	Specify the font size to use in the Results window. The default is 10.
Foreground	Select a color for the foreground from the drop list provided.
Background	Select a color for the background from the drop list provided.
Bold	Check this box to embolden the color of the text.

Results tab

The **Results tab** allows you to configure how results display.

Result Font	Specify the fonts to use in the Results window. The default is Arial.
Size	Specify the font size to use in the Results window. The default is 10.
Limit Result (rows)	Specify the default result limit you want to use. The system default is 100. You can override this setting by pressing the More button on the builder window and changing the value in the Limit field.

Compiler tab

The **Compliler tab** allows you to configure the local compiler (eclcc).

If you are running under Windows and want to compile locally, you need to install the Microsoft VS 2008 C++ compiler. (either Express or Professional edition) **. Linux users** need GCC.

An **Override Automatic Compiler Selection** checkbox is available should you want to override the default compiler selection, or location. Use this option if you are using multiple versions of the client tools or if you installed the compiler in a different location.

Compiler	Specify the compiler's full path and executable name.
Arguments	Optional compiler arguments. See the Command Line ECL section for more info.
Working Folder	Specify the full path to the folder you want to use when compiling locally.
WU Arguments	Optional command line parameters to pass to a locally compiling workunit.
ECL Folder	Specify the full path to the folder you will use to store your ECL files. Some folders are automatically predefined when ECL IDE is installed. They are used to store ECL sample and example code. The Arrows next to the ECL Folder allow you to move folders up and down in appearance in the workspace
Reset	Resets the values of all fields to the default settings.

Other tab

The **Other tab** allows you to configure other settings.

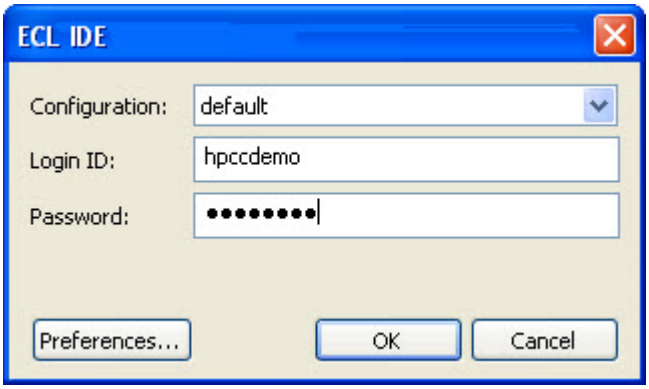
Active Workunits Refresh Interval (secs)	Specify the refresh interval (in seconds). This determines how often the Active Workunit Toolbox refreshes. The default is 10.
Test GPF	For internal use only.
Disable Invoke	Check this box to disable Invoke.
Graph Search Includes Tooltips	Check this box to ensure that tooltips are also included in your search.
Ignore Whitespace on Compare	Check this box to ignore whitespace (ii.e. spaces and blank lines) when comparing ECL files.
Show CR/LF on Compare	Check this box to show the type of line endings in the compare window when comparing two ECL files (for example, on checkin or history).
Ignore Server Version	For internal use only.
WU Fetch Limit	The maximum number of WUs that can be fetched, for example, when looking for workunits sent during a specific year/month etc.
ECL Watch Graph Control	While the graph viewer control is built in to ECL IDE, ECL Watch requires this control to be installed so that it can display graphs properly. The currently installed version is identified and the version number of the new version shown. You should only adjust this setting if instructed to do so by customer support.

Login

When you start the ECL IDE, the Login window displays.

- If you have created a configuration file to connect to the desired HPCC environment, select it from the **Configurations** drop list. If not, press the **Preferences**button to create one.
- Enter your credentials (Login ID and Password) and press the **OK**button.

  The Login window displays errors (e.g., invalid login information) if it cannot connect properly as shown below.

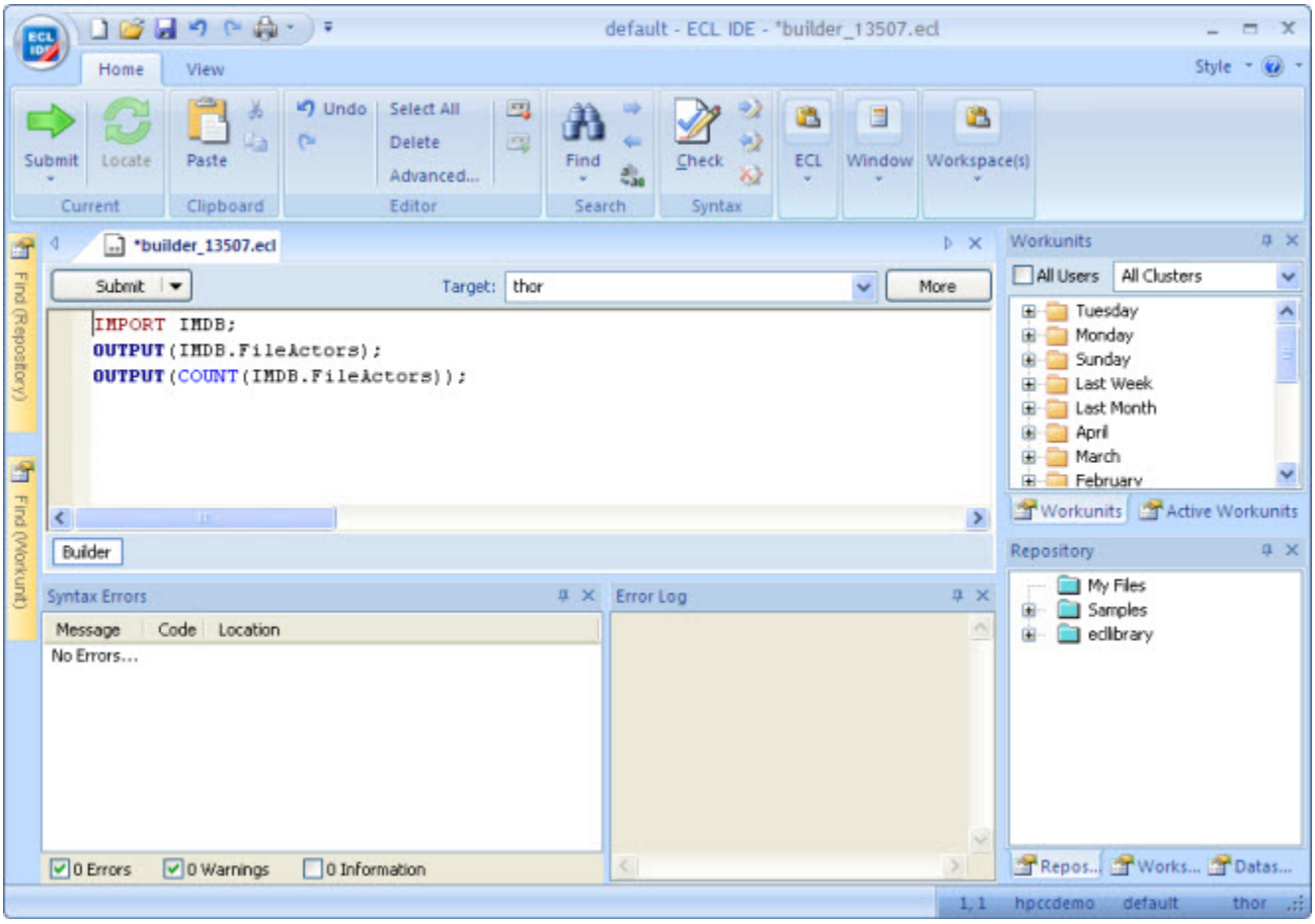


	The title bar of the login window also displays the version of the ECL IDE.

The ECL IDE Workspace

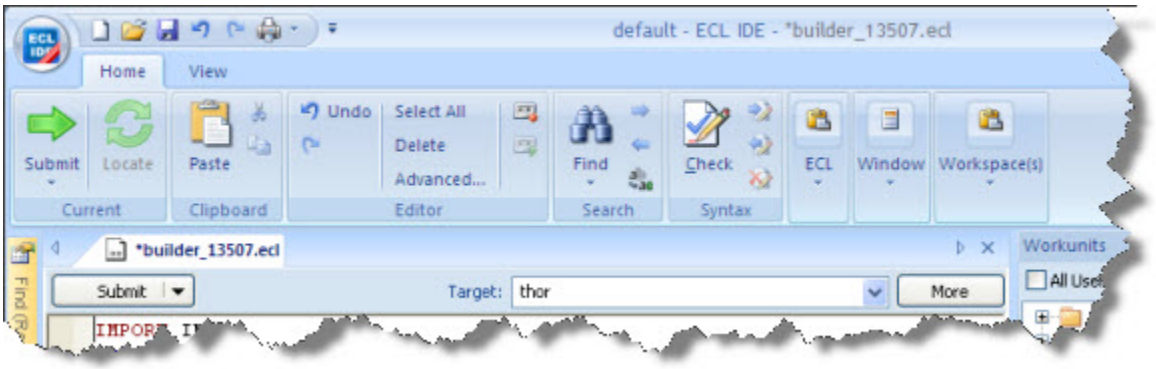
The ECL IDE workspace is customizable. You can choose which toolboxes to display and the location to dock them. You can also move toolbars to the position you choose. In addition, you can open child windows in normal mode or maximized or minimize the Ribbon.

In this manual, you will see screen representations using the locations of toolboxes pinned right.



Using the Ribbon

The Ribbon makes the commands that you need use to complete a task easy to find.



Commands are organized in similar groups contained in a tab. Each tab contains commands that relates to a type of activity, such as Syntax Checking or Search.

The Ribbon reduces clutter by showing some tabs only when needed. For example, the Browser Navigation tab is shown only when ECL Watch is active.

	After you have learned the locations of commonly used commands, you can gain more available space by minimizing the Ribbon.

To minimize the Ribbon:

- Press the  button on the right side of the **Quick Access Toolbar.**

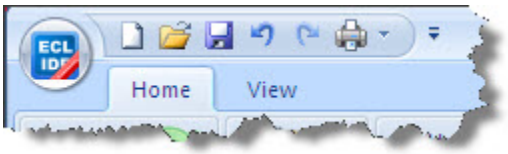
The menu displays,

- Select **Minimize the Ribbon**.

To use the Ribbon while minimized, CLICK on a tab, then CLICK the option or command you want.

The Quick Access Toolbar

Quick Access toolbar provides shortcut buttons for commonly used tasks, such as New Builder Window, Save, or Print. You can customize this button to add other tasks or remove buttons you don't want on the toolbar. You can also choose to display it beneath the Ribbon.



To customize the Quick Access Toolbar:

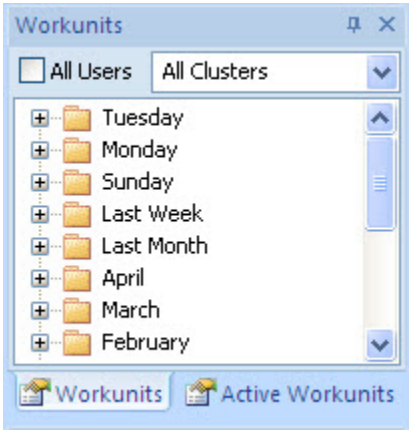
- Press the  button on the right side of the **Quick Access Toolbar.**
- Use menu commands to customize.

Working with Toolboxes

Toolboxes can be docked, pinned, unpinned, grouped, or hidden. In the graphic shown above, all toolboxes are shown pinned to the right (a pre-configured options on the View ribbon tab).

Grouping, Docking, and AutoHide

Once grouped, the toolboxes open and close as a group until ungrouped. AutoHide mode (unpinned state) also acts upon the group.



In the upper right corner, two toolboxes (Workunits and Active Workunits) are shown grouped. Navigate between grouped toolboxes by clicking on the tab for the toolbox you want to display.

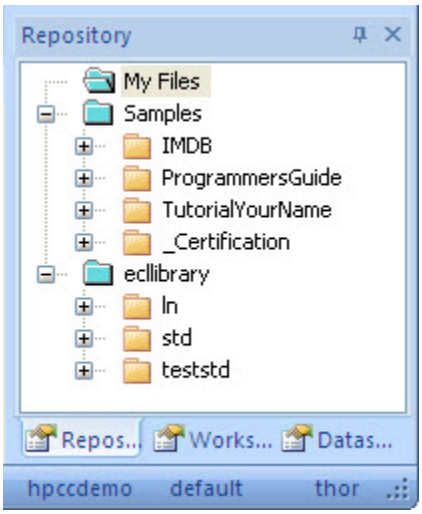
To group toolboxes, drag a toolbox and drop it on another toolbox. To ungroup, drag a toolboxes tab and drop it in another area. To move a group, drag it by the title bar.

To autohide (unpin) a toolbox or toolbox group, press the pushpin at the top of the toolbox. In autohide mode, the toolbox is hidden until you place the mouse cursor over its tab.

To reset toolbox locations, select the desired option from the **Reset Docked Windows** sub-menu on the View menu.

Repository Toolbox

This toolbox displays the ECL Folders for the current configuration. ECL files are organized by folders displayed in a tree format. Expand any folder branch to see the ECL files within it.



In addition to folders, plug-in service libraries are also displayed at the bottom of the repository. These special folders are pre-compiled libraries of functions that you can use in your ECL code. For example, the Standard Library (stringlib) contains common functions to operate on STRING fields such as the **ToUpperCase** function which converts characters in a string to uppercase.

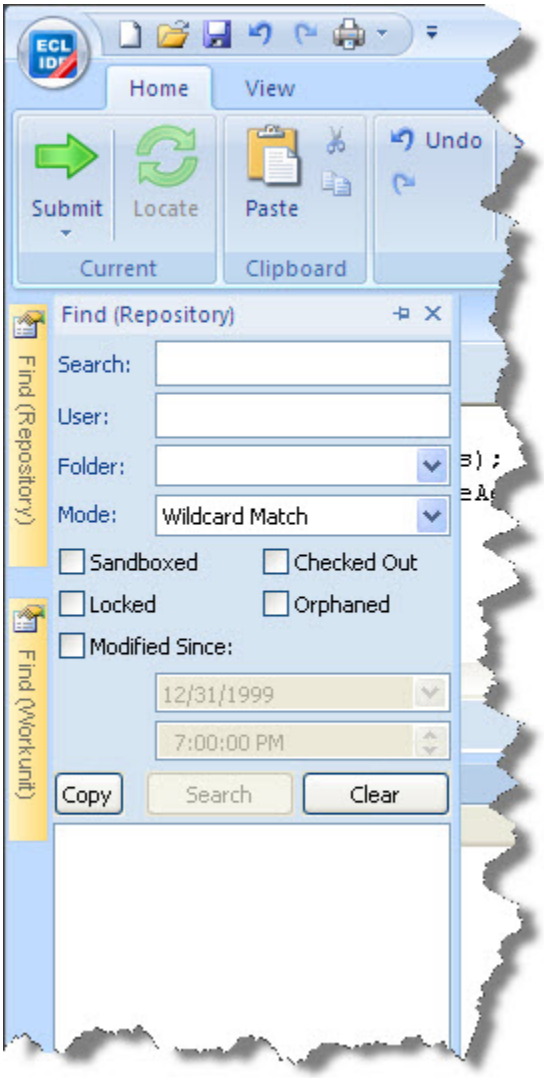
Plug-in service libraries are added to a system in one of three ways. Several are provided as standard. Others are available as optional addons. And, you can create your own and add them to a system.

A right-click on the folders in the Repository Toolbox gives you an option of **Locate File in Explorer** which will then open that folder in a new explorer window.

You can rearrange the order of the folders in the Repository Toolbox by modifying the order they appear in the compiler tab of the preferences window.

Find (Repository) Toolbox

This toolbox allows you to search the active Repository. Results are organized by folder and displayed in a tree format. Expand any folder to see the ECL files within it.



Search	Type your search phrase or regular expression (depending on the mode selected).
User	Limits search to a single user.
Folder	Limits search to a single folder.
Mode	Select the search mode from the drop list. There are several options.
	Wildcard match: allows a standard search and supports the use of ? to represent any single character or * to represent any number of characters.
	Regular Expression: allows a regular expression search.
Sandboxed	Check this box to only locate ECL files which are in your sandbox.
Check out	Check this box to only find ECL files that have been checked out.
Locked	Check this box to only find ECL files that have been locked.

Orphaned	Check this box to only find ECL files that have been orphaned.
Modified since	If checked, this option enables the date and time fields. Enter the time and/or date limit (or press the lookup button to select the date from a calendar). This filters ECL files based on the date and time supplied.
Copy	Press this button after the search to copy the list of found ECL files.
Search	Press this button to initiate the search. When completed, search results are displayed at the bottom of the toolbox.
Clear	Press this button to clear the search results.

WorkunitsToolbox

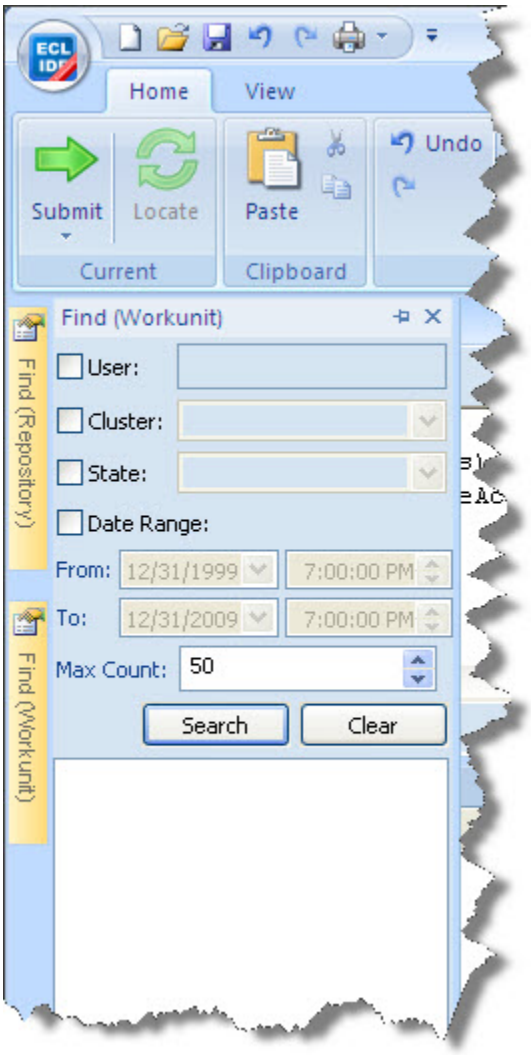
This toolbox displays Workunits for the environment to which you are connected. Workunits are organized into "folders" by date and displayed in a tree format. Expand any branch to see the Workunits within it.

Active Workunits

This toolbox displays active Workunits for the environment to which you are connected. Active Workunits are those which are in one of the following states: submitted, compiled, running, blocked, or aborting. The display is automatically refreshed based on the interval specified in the **Preferences** window.

Find (Workunits) Toolbox

This toolbox allows you to search online Workunits in the environment to which you are connected.



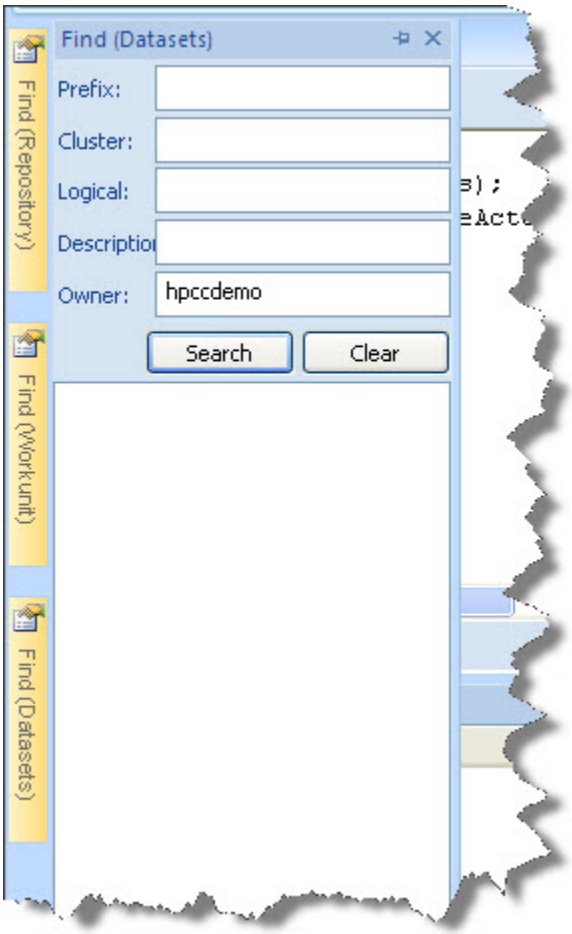
User	If enabled, this filters on the username. Enter a user's name to see only that user's Workunits.
Cluster	If enabled, select a cluster from the drop list to see only that cluster's Workunits.
State	If enabled, select a State from the drop list to see Workunits with that state.
Date Range	If enabled, this allows you to specify a range of dates/times to limit the display results
Max Count	Limits the number of Workunits to display.
Search	Press this button to initiate the search. When completed, search results are displayed at the bottom of the toolbox.
Clear	Press this button to clear the search results.

Datasets Toolbox

This toolbox displays Logical files for the environment to which you are connected. Logical files are are organized into “folders” by scope in which they were created and displayed in a tree format. Expand any branch to see the files within it. Right-click to copy the label of the file to paste into your code.

Find (Datasets) Toolbox

This toolbox allows you to search for Logical files in the environment to which you are connected.

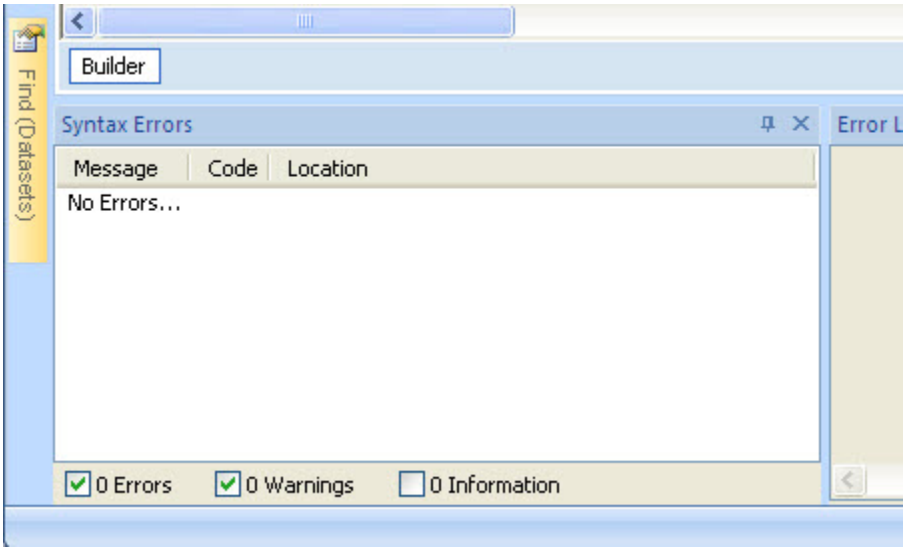


Prefix	This filters on the logical file's prefix.
Cluster	This filters on a cluster. Specify a cluster name from the environment.
Logical	Specify all or part of a file's logical name. Use an asterisk to wildcard a name (for example, *person* find any file containing person in its logical name.
Description	Specify all or part of a file's description. Use an asterisk to wildcard a description (for example, *person* find any file containing person in its description.
Owner	Specify an Owner name to find only files created by a specific user.

Syntax Errors Toolbox

The Syntax Errors toolbox displays syntax errors and warnings if any are found when conducting a Syntax Check. A syntax check is also performed when submitting a job (pressing Submit or Go button).

For example, referencing an ECL file which has not yet been defined will display a syntax error as shown below.

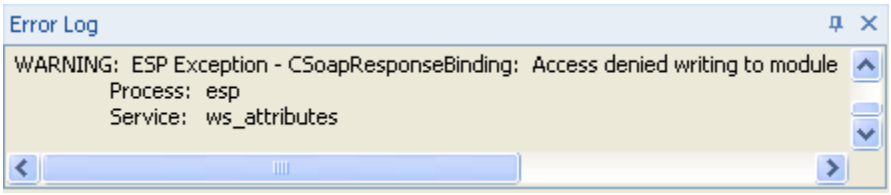


DOUBLE-CLICK on any error displayed to take you to the line containing the error. If the error is in a different attribute, a DOUBLE-CLICK on the error opens that attribute in an editor window.

Error Log Toolbox

The Error Log toolbox displays errors and warnings from the HPCC system.

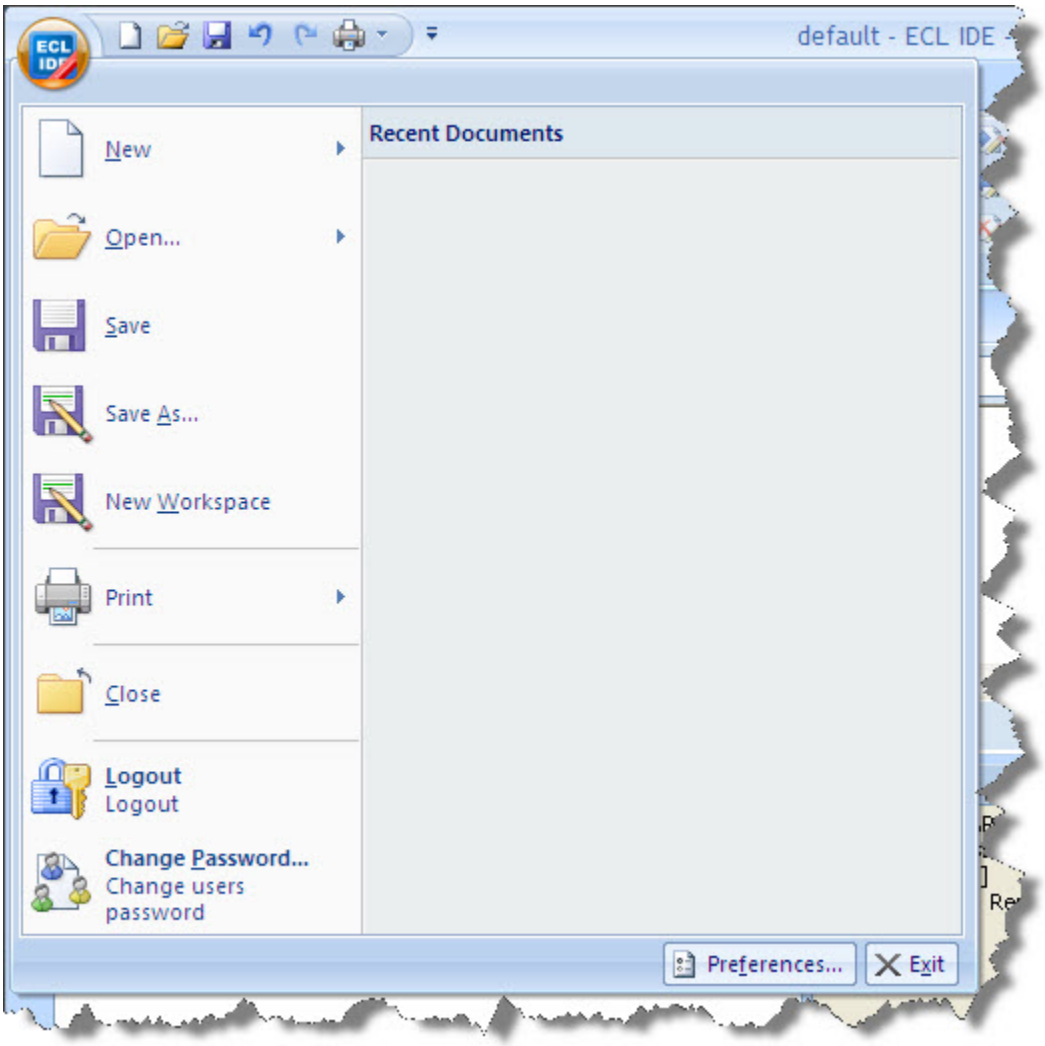
For example, attempting to create an ECL file in a folder without WRITE permission will display an access denied error as shown below:



You can copy the error message text using the right-click pop up menu option.

Orb Menu

Orb Menu



The Orb button menu provides a list of basic actions enabling you to create and gain access to files and workspaces. It also supplies some basic administrative functions which can be used to control and specify access details for the environment(s) you use. It contains the following selections:

New	opens a new ECL Builder window. CTRL-N
Open	Provides access to the following Open submenus: **Open Builder:** opens a saved query (.ECL file) into a new ECL Builder window. CTRL-O **Open Attribute:** opens a dialog to name the attribute to open. Enter the fully qualified attribute name, then press the **OK** button. **Open Workunit:** opens a dialog to name the Workunit to open. Enter the Workunit ID (WUID), then press the **OK** button.
Save	saves the contents of the window with input focus. An ECL Builder window query is saved to an .ECL file. An Editor window saves the ECL file to the Repository. CTRL-S
Save As	saves the contents of the window with input focus to a new filename.

Print	Provides access to the following Print submenus: **Print Setup:** opens a page setup dialog where you can set margins, paper size, and orientation, etc. **Print:** prints the contents of the active window. CTRL-P
Close	closes the active window. CTRL-F4
Logout	closes the session and allows you to login again.
Change Password	allows you to change your password.
Exit	closes the ECL IDE.
Preferences	opens the Preferences window to allow you to configure ECL IDE options.

Ribbon Tabs

Current Tab

The **Current Tab** contains the following selections:

	**Submit**	Submits the ECL code in the active Builder window. CTRL-ENTER
	**Locate**	Locates the ECL file selected in a builder or editor window. F9

Clipboard Tab

The **Clipboard Tab** contains the following selections:

	**Paste**	Standard Windows text paste operation. CTRL-V
	**Cut**	Standard Windows text cut operation. CTRL-X
	**Copy**	Standard Windows text copy operation. CTRL-C

Editor Tab

The Editor Tab contains the following selections:

	**Undo**	Restores the text to state it was in before the last change (multiple Undo calls will restore each successive change back until the text is in its original format). CTRL-Z
	**Redo**	Restores the text to state it was in before the last Undo (multiple Redo calls will restore each successive change back until the text is in its final changed format). CTRL-Y

Select All Highlights all text in the currently active window. CTRL-A

Delete Removes selected text from the currently active window delete.

Record\Stop Record macros for playback later. CTRL-SHIFT-R

Play Playback recorded macro**.** CTRL-SHIFT-P

Advanced... The Advanced editor options are as follows:

Complete Word	A list of keywords and folders from which you can select. This allows you to complete a word by selecting from the list. CTRL-Space
Match Brace	When the insertion point is adjacent to a grouping character (parenthesis, square bracket, or curly brace) moves the insertion point to the matching grouping character, beginning or end. CTRL-E
Select to Brace	When the insertion point is adjacent to a grouping character (parenthesis, square bracket, or curly brace) selects to the matching grouping character. CTRL-SHIFT-E
Toggle Comment	Toggles comment characters before selected lines. CTRL-Q
Box Comment	Encloses selected text in a comment structure. CTRL-SHIFT-Q
Uppercase	Changes case of selected text to uppercase. CTRL-SHIFT-U
Lowercase	Changes case of selected text to lowercase. CTRL-U
Line Numbers	Toggles line numbers in the editor. CTRL-L
Word Wrap	Toggles word wrapping in the editor. CTRL-W
Set Tab Size	Select to set a tab width of between 1-16.

Search Tab

The **SearchTab** contains the following selections:

Find Locates specific text in the currently active window. CTRL-F

Find Next Locates the next instance of the same specific text in the currently active window. F3

Find Previous Locates the previous instance of the same specific text in the currently active window. CTRL-F3

Replace Locates specific text in the currently active window and replaces it with new text. CTRL-H

Syntax Tab

The **SyntaxTab** contains the following selections:

Check Syntax	Checks the ECL code in the window with input focus for syntax errors.
Next Error	Moves the insertion point to the next error and changes the error message displayed.
Previous Error	Moves the insertion point to the previous error and changes the error message displayed.
Clear Errors	Clears the errors displayed in the Syntax Errors toolbox.

ECL Tab

The **ECL Tab** contains the following selections:

Show Complexity...	displays a complexity score. Use this option to evaluate and optimize queries.
Show Dependency...	displays a list of ECL files upon which the current ECL file depends.
Copy Title	copies the title of the active tab to the clipboard.
Locate Current	highlights the current ECL file in the tree
Locate Selected	highlights an item in the Repository tree when its name is highlighted in a text control.
Open Selected	opens an item in the Builder window when its name is highlighted in a text control.

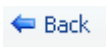
Browser Tab

The **Browser** tab is shown only when the ECL Watch page is displayed for a workunit. To view the ECL Watch page from within ECL IDE, simply click on the workunit ID during or after it has completed running. ECL Watch is one of the pages that can be viewed at this point. The Browser tab is shown at the top of the Builder window and contains the following selections:

Stop	Stops the web page from further processing.

Refresh	Reloads the current page to make sure you have the latest version.

Launch New Window	Opens the current page in a new browser window.



Back	Returns to the last page viewed.

Forward	Moves forward to the next page (only available after having moved back).

Window Tab

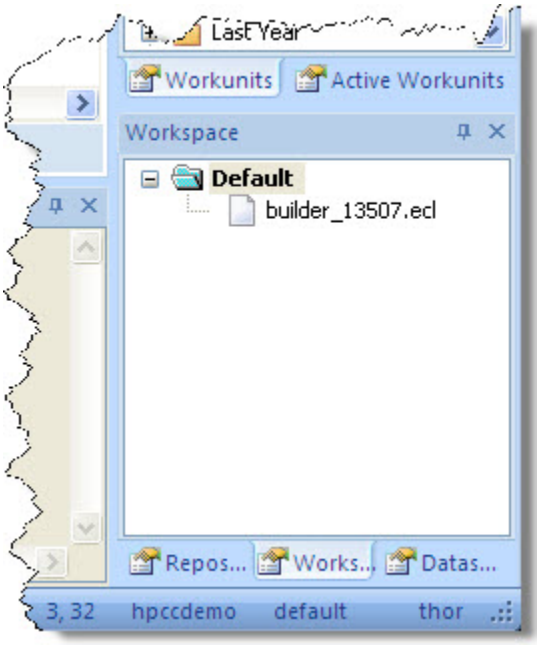
The **Window** Tab contains general actions for controlling open windows on the desktop.

Cascade	displays all open windows so that their top left corner is visible from beneath the window on top.
Tile	displays all open windows one above the other
Arrange Icons	lines up icons for all minimized windows
Sort	sorts open windows
Group	groups similar windows together
Degroup	de-groups windows previously grouped

As well as these options, there is also a Windows drop menu on this tab. These are provided to help you control specific windows which are open on the desktop. All currently open windows are listed on this menu and you can switch to any window listed by simply clicking on the one you want to view. Other available selections are:

New Builder	Opens a new, empty builder window.
Close All	Closes all open windows.
Windows	Displays a dialog listing all currently open windows. Simply click on one or more windows to highlight them in the list and press the required button. Activate, displays the window(s) you selected, OK takes you back to where you were, Save, enables you to save the selected window(s), and Close Window(s) closes only those windows highlighted in the list.

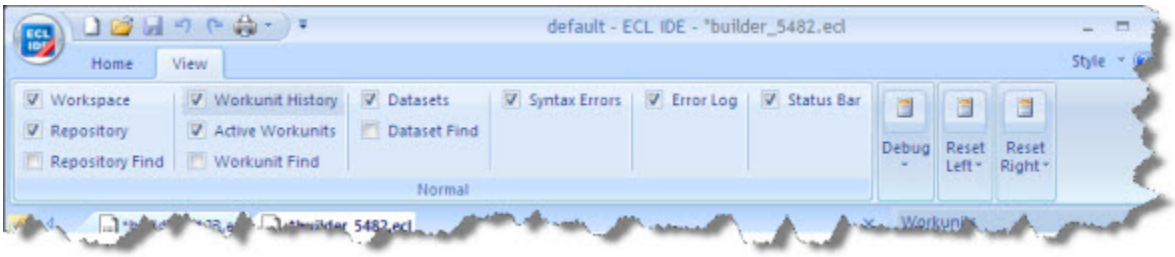
Workspace(s) Tab



The **Workspace Tab** tab allows you to preserve the workspace you are using for future use or remove an existing workspace that you no longer need. Once you have organized a workspace, use **Add** to save it. Press **Remove** to delete the workspace you are currently using.

Your saved workspace(s) are shown in the list under the **Default** option. Selecting a workspace as the default will ensure that every time ECL IDE is started, the selected workspace is automatically loaded ready for you to use.

View Tab



The **View** tab allows you to choose which toolboxes to view. In addition, it has options to reset the positions of these toolboxes to any of the predefined locations.

Help Menu

The **Help** menu contains the following selections:

Download Resources	opens the Download Resources page of ECL Watch in a browser
About	displays information about the program and the server

Pop-up Menus

Workunit Pop-up Menu

Right-click on any Workunit for a Pop-up menu containing:

Open Results	opens an ECL Builder window displaying the results tab
Show Graph	displays the graph for the selected workunit
Browse ECL Watch	opens a web page of the ECL Watch tool
View Exceptions	displays the contents of the Exceptions tab for the selected workunit.
Open ECL	opens an ECL Builder window displaying the ECL code for the Workunit
Save as Excel Workbook	saves workunit results to an Excel workbook. Each result set is saved as a sheet in the workbook.
Copy WUID	copies the Workunit ID to the clipboard
Abort	stops an executing Workunit
Resubmit	reruns the Workunit
Delete	removes the Workunit from the system data store
Protect	protects the Workunit (prevents deletion and archiving)
Unprotect	removes protection from the Workunit
Follow Up	enables you to mark the selected workunit so you can come back to it later. The **Flag** option sets the marker and **Clear** removes a flag that was previously set.
SetPriority	allows you to set the priority in the queue for a running Workunit. You can move a workunit up or down one place in the queue, or send a workunit to the top or bottom of the queue.
SetState	allows you to set the state of a Workunit.
Refresh	refreshes the display

Repository Pop-up Menu

Right-click in the **Repository Window** for a Pop-up menu containing options which are available depending on whether you have a folder or ECL file selected.

Open in Builder Window	Opens the selected ECL file in an ECL Builder window
Locate File in Explorer	Only valid in locally compiled workunits
Copy "<attribute_name>"	Copies the fully qualified ECL file name to the clipboard
Paste Selection	Pastes the ECL file name you copied where the cursor is currently positioned
Check Syntax	Displays a sub-menu of all available INVOKE structures for the selected ECL file
Invoke	Displays a sub-menu of all available INVOKE structures for the selected ECL file
Insert Root Module	Adds a new folder at the top most level of the repository tree
Insert Module	Adds a new ‘nested’ folder within the currently selected folder
Rename Module	Renames the selected folder.
Delete Module	Deletes the selected folder.
Insert Attribute	Adds a new ECL file inside the selected folder.
Rename Attribute	Renames the selected ECL file

Copy Attribute	Enables you to copy the selected ECL file so it can be pasted into a different folder.
Move Attribute	Moves the selected ECL file to specified folder
Delete Attribute	Moves the selected ECL file to the Trash folder on your desktop
Refresh	Refreshes the repository folder display

Results Pop-up Menu

Right-click on Results Pop-up menu containing:

Save	saves the results to a file
SaveAs	saves the results to a new file
Save As XML	saves the results to a file in XML format
Cut	removes (from display) and copies the selected results to the clipboard
Copy	copies the selected results to the clipboard
Paste	pastes contents of clipboard into result.
Copy As ECL	copies the selected results to the clipboard in ECL format
Send to “”	sends the selected results to the application associated with CSV files (for example, Microsoft Excel)
Select All	selects all results
Select Row	selects the current row
Select Column	selects the current column
Delete Selection	deletes the selected results from displayed results
Show Column in Hex	displays hex value for the current column
Format	displays a submenu with options to format selected cells.
Auto Size Column	adjusts the current column width to allow all characters to display
Auto Size All Columns	adjusts the all column widths to allow all characters to display
Find	locates specific text in the results
Find Next	locates the next instance of the same specific text in the results
Find Previous	locates the previous instance of the same specific text in the results
GoTo	positions the cursor at a specified row number in the results
Drilldown	sends the selected result rows to a previously defined Drilldown ECL file. This method sends the selected rows to an ECL file as a set.
Drilldown (1 per row)	sends the selected result rows to a previously defined Drilldown ECL file. This method sends each row to the Drilldown separately.
Drilldown II	generates ECL code that ends with a call to the selected MACRO ECL file.
Copy WUID	copies the Workunit ID to the clipboard
Save As Excel Workbook	sends the result set to a Excel Workbook with each result set on a sheet.
Sync Workunit	locates the current Workunit in the Workunit History tree
Close Workunit	closes the current Workunit
Delete Workunit	deletes the current Workunit

Using Drilldown

There are three different versions of drilldown: Drilldown, Drilldown (1 per row), and Drilldown II.

Drilldown sends the selected result row(s) to the selected MACRO ECL file as a single string of XML data.

Each selected row is contained within <row></row> XML tags. The MACRO must use LOADXML to instantiate an XML scope and should use #FOR(row) to process each separate record. Each field in the record is treated as a template symbol and may be referenced in the MACRO as %fieldname%.

Example Drilldown MACRO:

```
export TestDrilldown(xmlRow) := MACRO
LOADXML(xmlRow);
#DECLARE(Ctr)
#SET(Ctr, 0 )
#DECLARE(OutputStr)
#SET(OutputStr, 'OUTPUT(ProgGuide.Person.file(PersonID IN ['
          )
#FOR(row)
#IF(%Ctr% = 0)
#APPEND(OutputStr, %'personid'% )
#SET(Ctr, 1 )
#ELSE
#APPEND(OutputStr,',' + %'personid'% )
#END
#END
#APPEND(OutputStr,']));\n')
%OutputStr%
ENDMACRO;
```

Drilldown (1 per row) sends the selected result row(s) to the selected MACRO ECL file as XML data. If multiple rows are selected, each is sent to a separate MACRO call as a single string of XML data. The MACRO must use LOADXML to instantiate an XML scope. Each field in the record is treated as a template symbol and may be referenced in the MACRO as %fieldname%.

Example Drilldown (1 Per Row) MACRO:

```
export TestDrilldownOnePerRow(xmlRow) := MACRO
LOADXML(xmlRow);
OUTPUT(ProgGuide.Person.file(PersonID = %personid%));
ENDMACRO;
```

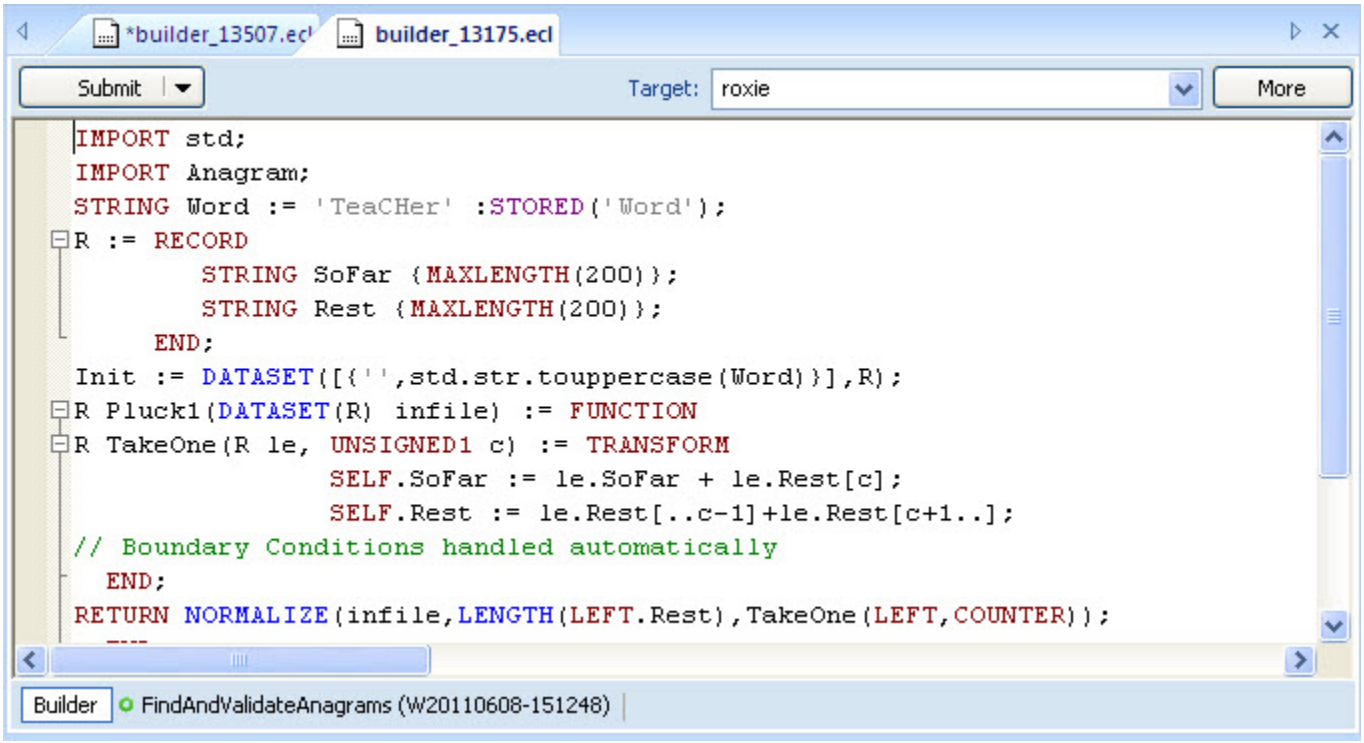
Drilldown II generates ECL code that ends with a call to the selected MACRO ECL file. Each field in the result is defined as a separate set ECL file and as a field in an inline DATASET. The MACRO should be written to make use of whichever of the ECL files best allows it to accomplish its task (typically, to show all base data from which the result set was drawn).

Example Drilldown II MACRO:

```
EXPORT TestDrilldownII := MACRO
OUTPUT(ProgGuide.Person.file(personid IN
          Set_personID));
ENDMACRO;
```

ECL Builder Window

This window allows you to edit, build, and run ECL Queries. It also shows results and Workunit information for each Workunit created by submitting the ECL code.



The **More** button displays advanced options. See the Advanced Builder Window Options section below for details. An asterisk on the button indicates that advanced options have been set.

The tabs beneath the text area represent Workunits created by submitting the ECL code. Select a Workunit tab to see results, graphs, and the ECL watch page for the Workunit.

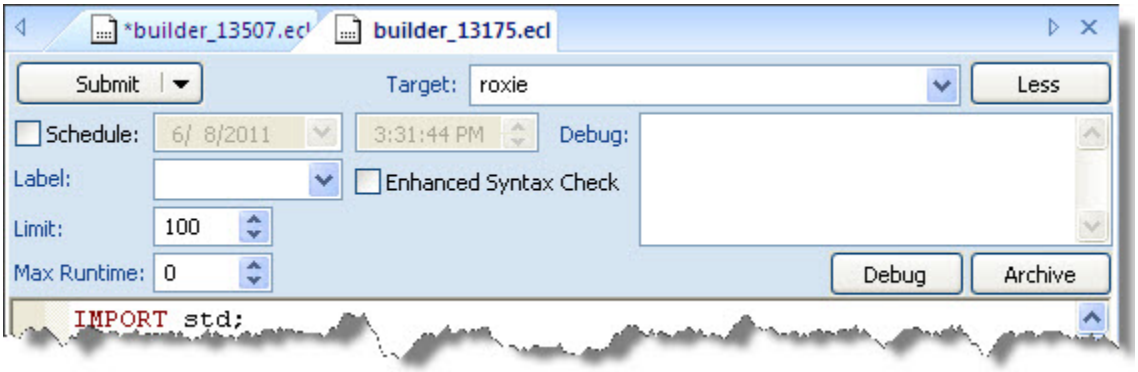
You can save your queries to an ECL file by pressing the **Save** button on the toolbar, or selecting the **File > Save** menu selection.

Using CTRL+MOUSEWHEEL will grow or shrink the font size used in the window.

There is also a right-click pop-up menu with the all the options found on the **Edit** menu.

Advanced Builder Window Options

Press the **More** button to display the advanced options. Press the **Less** button to close this area.



The **Submit** button submits the workunit. Select the options you want to use when running the workunit as follows:

The **Target** drop list shows the cluster where the query will be sent. Select Local, if you want to run the query locally on your computer.

The **Schedule** checkbox allows you to schedule queries to run at a specific time. When enabled, the **Submit** button becomes a **Schedule** button. The **Schedule** button sends a scheduled query to the HPCC.

When the job is submitted it is assigned a workunit ID and appears in the workunit list as *scheduled* (with a clock icon). You can monitor the workunit in the same manner as a submitted workunit.

The **Label** drop list allows you to select a Repository Label which specifies a previous version of ECL files.

The **Limit** spinbox allows you to override the default setting for result limits.

The **Max Runtime** spinbox allows you to specify a timeout that will abort the job if exceeded.

The **Debug** button invokes the browser which enables you to step through your ECL to monitor progress and identify problems.

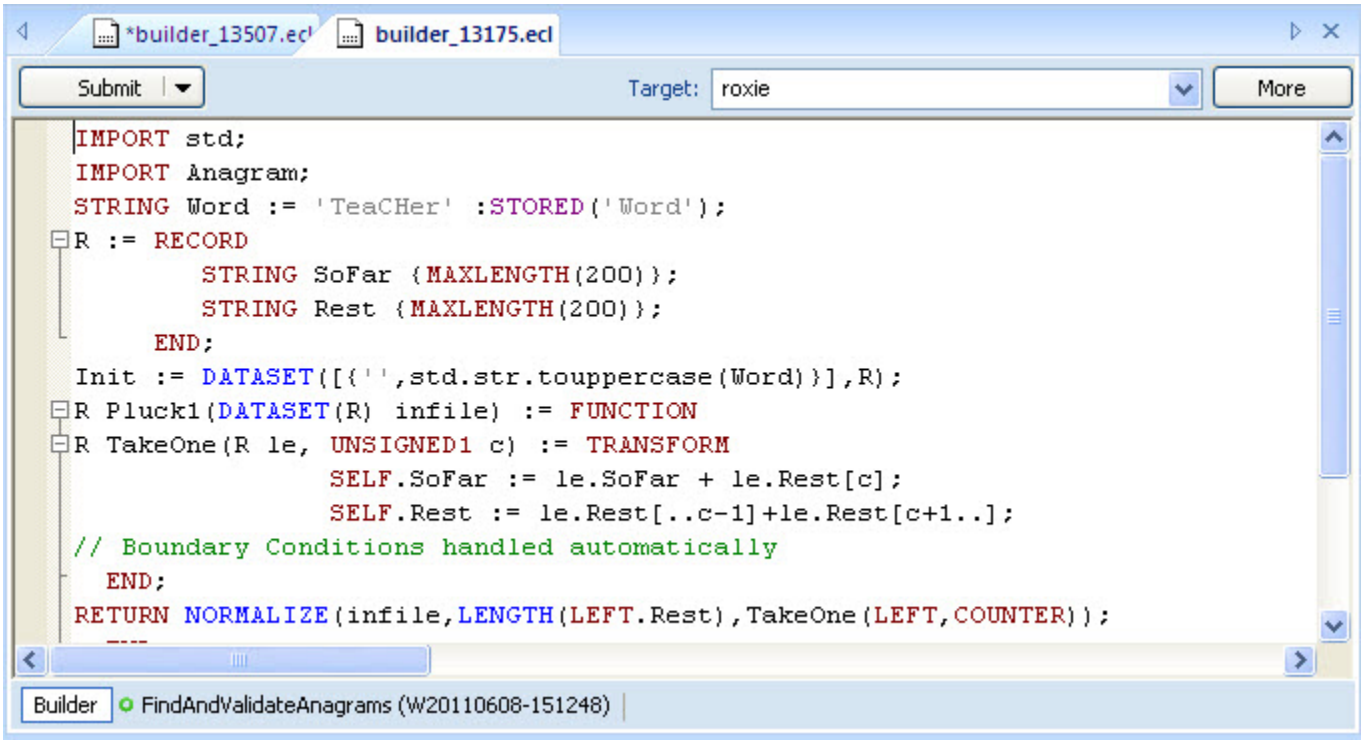
The **Archive** button is for debugging purposes and archives all ECL (including dependencies) into the workunit for examination without executing the code.

The **Enhanced Syntax Check** checkbox allows you to enable enhanced syntax checking.

The **Debug** textbox allows you to supply debug options (with our assistance) to help our developers resolve issues.

Editing

The Builder Window also supports editing ECL files.



Using CTRL+MOUSEWHEEL will grow or shrink the font size used in the window.

There is also a right-click pop-up menu with the all the options found on the **Edit** menu.

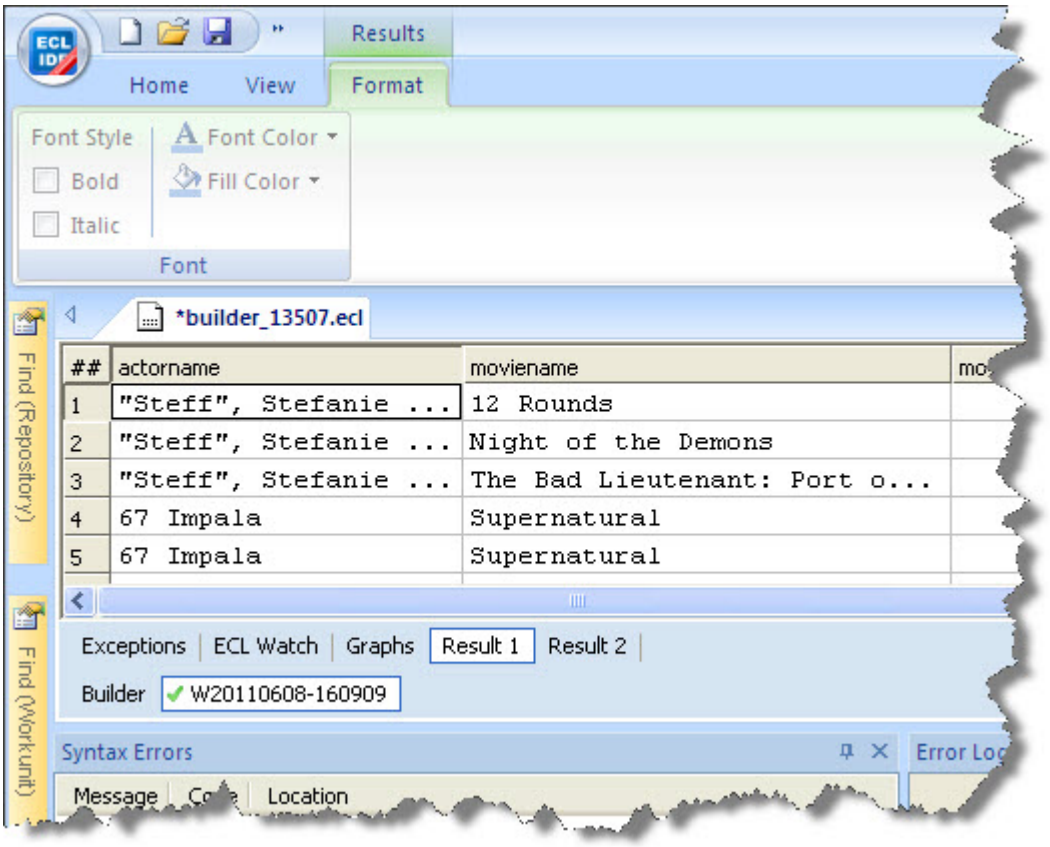
Results Viewer

Each result is displayed on a results tab on the workunit tab.

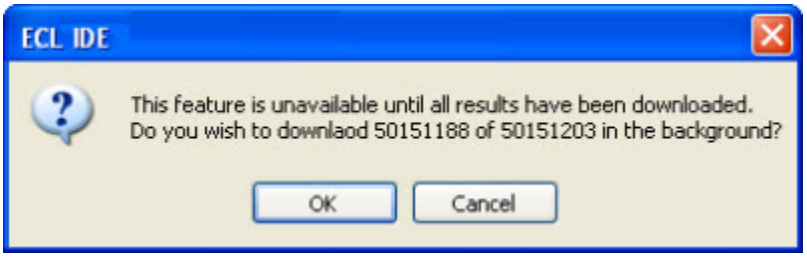
The results window display can be configured in Preferences. Additionally, you can modify a result list display in the following ways:

- CLICK on a column header to sort by that column.
- Use the Format menu or toolbar to modify selected cell font or background colors.
- Use Ribbon bar buttons or right-click the popup menu. Cut, Copy, Paste, or Delete to remove or add cells, rows, columns to what is displayed. This does not change the actual result, only the display.

- Autosize one column or all columns.



If you attempt an action that requires more data, such as sorting, you are asked to confirm before the download begins. This is particularly useful when large record sets are returned.

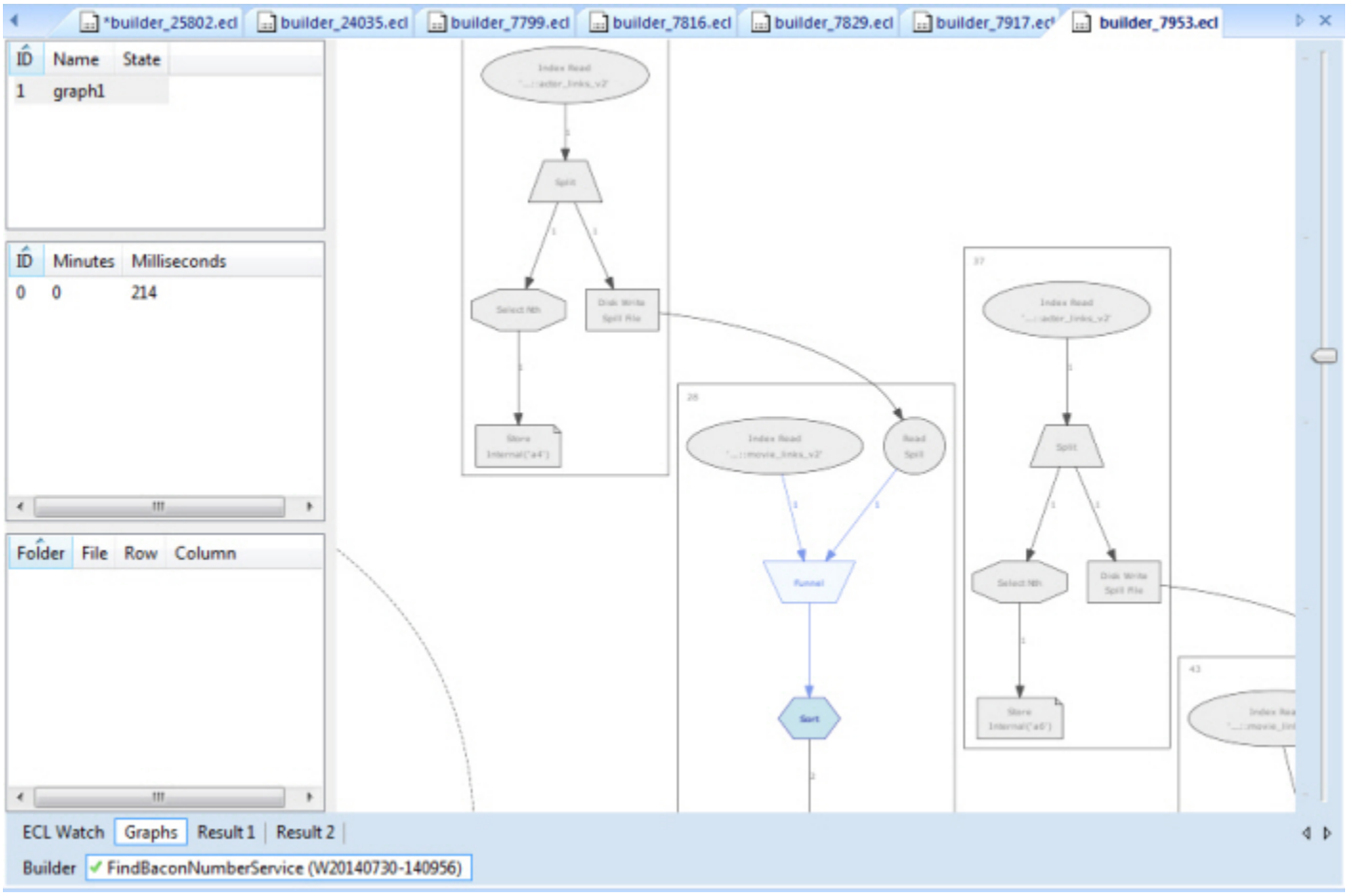


Graph Viewer

A workunit's execution graph is displayed in the Graph Tab.

Modify the display in the following ways:

- Use the Ribbon bar Navigation buttons to move around a graph.
- Use the Ribbon bar Zoom options To Fit /To Width to adjust how the graph is displayed.
- Use the Ribbon bar Running buttons to watch the progress of a running workunit, refreshing the graph as it runs. Options are provided so you can Follow Active or Minimise Inactive areas of the graph according to your preference .

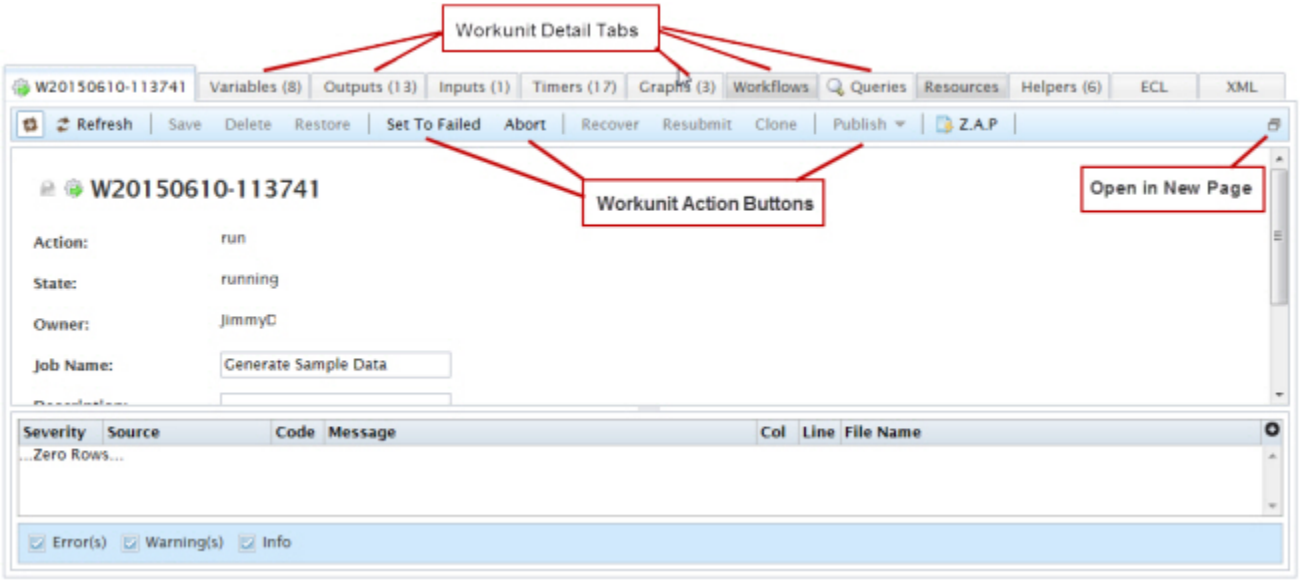


ECL Watch View

A workunit's Workunit Details page view is displayed in the ECL Watch Tab.

The Workunit Details page provides more information about a workunit.

You can see specific information about the workunit by selecting the various Workunit Detail tabs.



You can also perform actions on the workunit using the Workunit Action buttons.

	You can copy a Workunit's ECL Watch URL by rt-clicking on the **Open in New Page** icon and selecting **Copy Shortcut**.

- Press the **Refresh** button to refresh the workunit details.
- Press the **Save** button to save any changes to the workunit.
- Press the **Delete** button to delete the workunit.
- Press the **Restore** button to restore a workunit that has been archived by Sasha.
- Press the **Set To Failed** button to change the workunit state to failed.
- Press the **Abort** button to stop a running workunit and abort the job.
- Press the **Recover** button to restart the workunit.
- Press the **Resubmit** button to resubmit the job.
- Press the **Clone** button to clone the workunit.
- Press the **Publish** button to publish the workunit.
- Press the **Z.A.P.** button to package up system information for analysis. This is useful for bug reporting or troubleshooting.

ECL Debugger

Introduction

The ECL Debugger provides quick and easy access to features which show you exactly how your ECL query is working. It is integrated into the ECL IDE, giving immediate access to the debugging process as you develop your queries. It is a useful development tool helping you to:

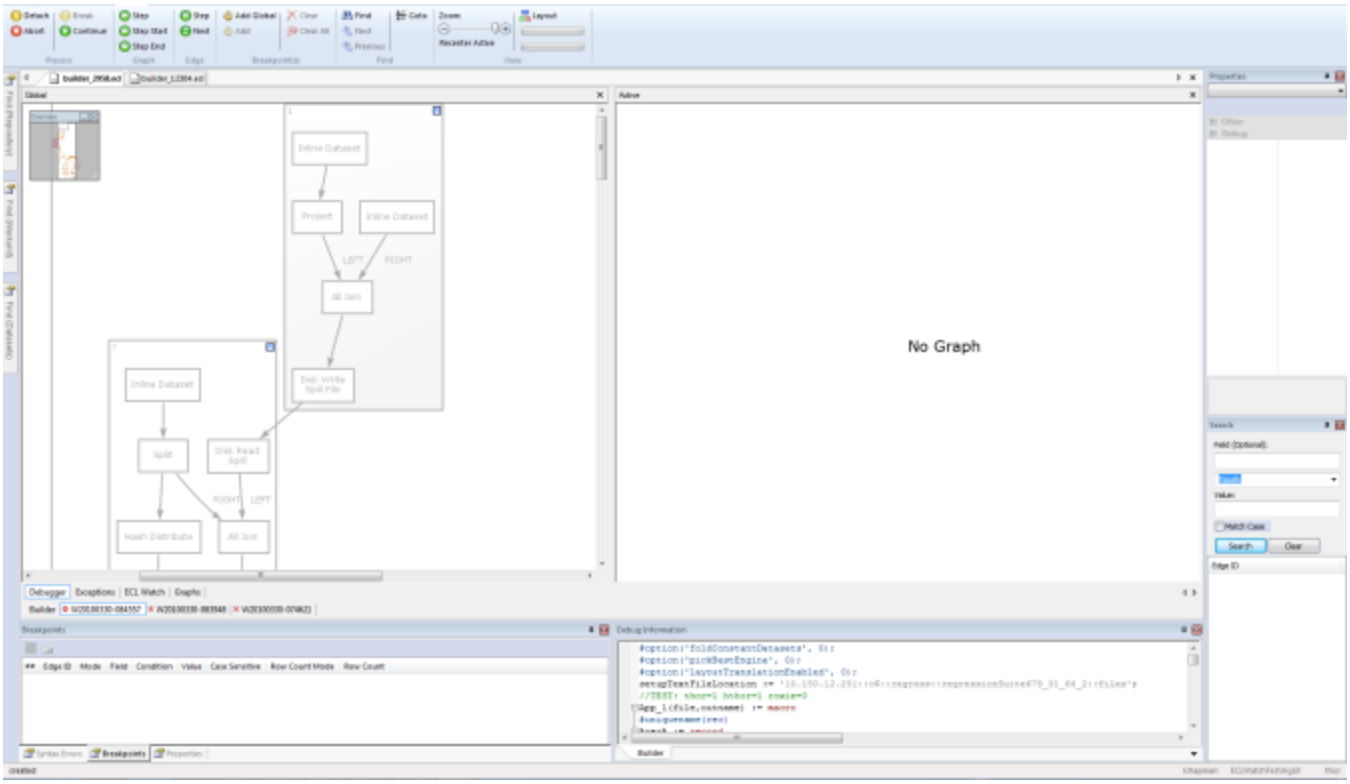
- Learn how the ECL Language works.
- Develop a query to see how it works.
- Verify that your query behaves as you expected.
- Identify why your query is not working as you expected.

Currently the debugger shows debugging information for any query which is run on an hThor or Roxie cluster.

Note: The ability to view debugging information for queries run against a Thor cluster is part of ongoing planned development of this tool.

Global and Active Graphs

The two graph windows are displayed side by side within the ECL IDE.



The Global Graph shows exactly what Roxie/hThor was asked to execute by the compiler. It is primarily used for navigating around the graph and setting breakpoints.

The Active window shows what is actually being executed. The graph in the Active window may change its shape during the debugging process.

These graphs may show different information. The Global Graph may not show the parts of the graph that cannot be executed, for example, IF statements or when using the Graph command. The Active window may show expanded sections for commands such as Graph or Library.

Edges and Vertices

The graphs are displayed as a series of edges and vertices.

A vertex is an ECL keyword or activity, for example, SORT or JOIN. Vertices are represented on the graphs surrounded by a rectangle. When a vertex is selected and has a definition associated with it, the ECL that caused it to be created is highlighted in the Debug Information window.

An edge is shown on the graphs as a line joining two vertices and represents the data flowing between them. When an edge is selected, the last N rows of data are displayed in the Debug Information window.

The Properties window shows the details associated with the currently selected edge or vertex, for example, the definition, id, label, name, etc.

Breakpoints

Using the ECL Debugger breakpoint feature, the debugging process can be paused at any point.

Global breakpoints stop the debugging process every time the specified condition is met. Local breakpoints stop the process only for that specific occurrence.

The debugging process pauses at the point at which the breakpoint is set. Breakpoints can only be set on edges. However, there are a number of ways to use breakpoints:

- A simple breakpoint does not contain any specific conditions. The debugging process pauses on reaching the edge where the breakpoint was set.
- A breakpoint containing conditions and a value, pauses the debugging process when that condition is met.
- A breakpoint set on the ECL code in the Debug Information window, pauses the debugging process when the first row of data arrives at the edge of the vertex you selected.

There are four breakpoint modes:

Break	Pauses the debugging process at the specified edge.
Skip	Skips any row that matches the conditions set.
Limit	Only allows the specified number of rows through.
Continue	Temporarily disables a breakpoint that is not currently needed but that might be required later.

Using the many different types of Condition settings available, means you can customize the debugging process to meet your specific requirements. For example, you may want to pause the debugging process when a specific field contains a specific value to make sure your query is behaving as you expect.

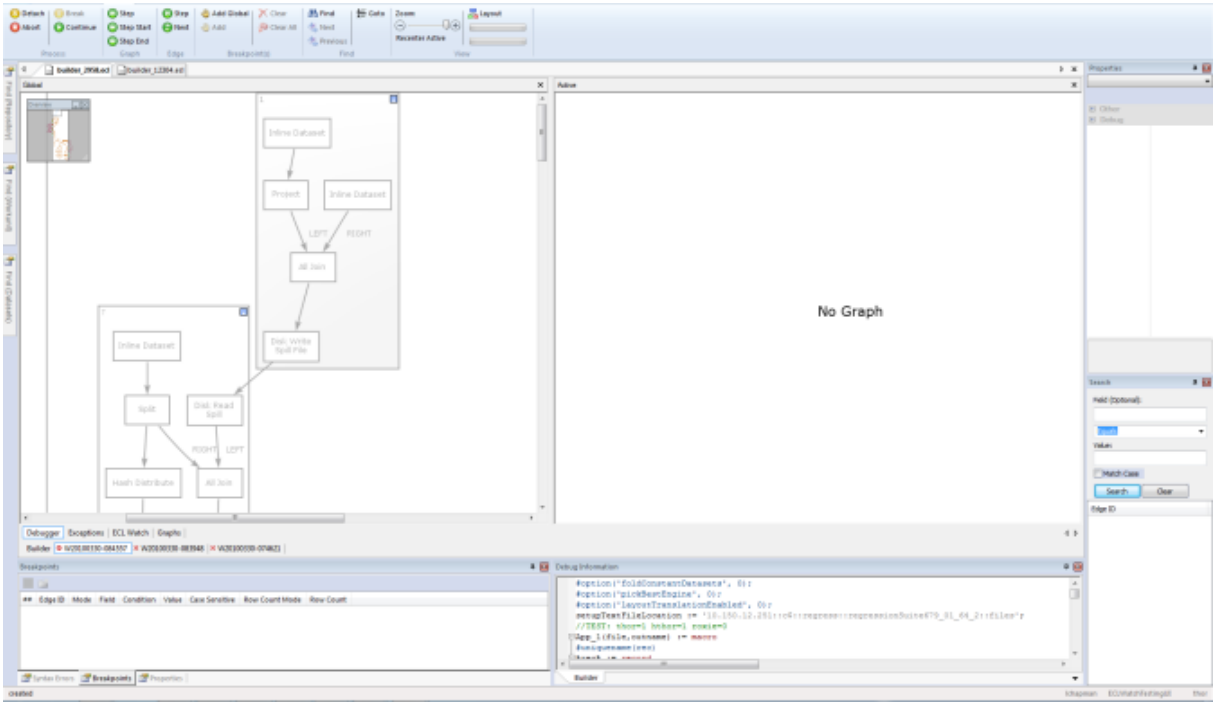
Using breakpoints helps you to understand how your query works enabling you to focus on those areas which are not working as you expected.

Each breakpoint you set on your graph is shown in the list in the Breakpoints window showing the settings used to create each one including, for example, Edge ID, Mode, Condition, etc.

Starting the ECL Debugger

- Login to the ECL IDE in the usual way.
- Open or create your ECL Query in a Builder window. Make sure that the Queue and Cluster you have selected point to either an hThor or a Roxie cluster.
- Press the More button located on at the top right of the Builder window. Press the **Debug** Button.

The ECL Debugger workspace displays a graph in the Global window, but the Active window is empty since the debug process has not yet begun.



Debugger Menus

The Home and View menu options are identical to those on the main the ECL IDE menu.

Basic Menu

The Basic ribbon bar contains the following features:

The **Process** options include:

Detach	Detaches from the current debugging session and allows the workunit to run to completion.
Abort	Aborts the current workunit which also causes the debugging session to abort.
Break	Pause the execution of the running workunit.
Continue	Continues the execution of a paused workunit

The **Graph** options include:

Step	Continues the debugging process until the next graph event. This is typically the start of a new active graph or the end of the current active graph.

Step Start	Continues the debugging process until a new active graph is about to start.
Step End	Continues the debugging process until the current active graph completes.

The **Edge** options include:

Step	Continues the debugging process until a new row of data is available on any edge in the active graph.
Next	Continues the debugging process until a new row of data arrives at the active edge or until the current active graph completes.

The **Breakpoint(s)** options include:

Add Global	Adds a global conditional breakpoint.
Add	Adds a local conditional breakpoint.
Clear	Clears all the breakpoints on the selected edge.
Clear All	Clears all breakpoints.

The **Find** options include:

Find	Locates a vertex with matching text in the graph you are currently using (Global or Active).
Next	Locates the next occurrence in the graph you are currently using.
Previous	Locates the previous occurrence in the graph you are currently using.
Goto	Locates the subgraph/vertex/edge which matches the specified ID and centers it.

The **View** options include:

Zoom	Resizes the graph according to your selection.
Recenter Active	When the graph is paused, clicking on this option takes you to the active edge on that graph.
Layout	Performs a new layout on the active graph. This can be particularly useful when the label have changed. The top progress bar shows that data is being sent/received from the server. The bottom progress bar shows when the graph layout is being calculated.

Advanced Menu

The Advanced ribbon bar contains the following features:

The **Global Filter** options include:

Running Only	Only shows items that have started to execute.
Search Results Only	Only shows items that have been found using the Search Window.

The **Active Filter** options include:

Running Only	Only shows items that have started to execute.
Search Results Only	Only shows items that have been found using the Search Window.

Stepping through a query

- At the start of the debugging process, Press Edge/Step. The graph will continue to the next row of data available at any edge of the active graph.

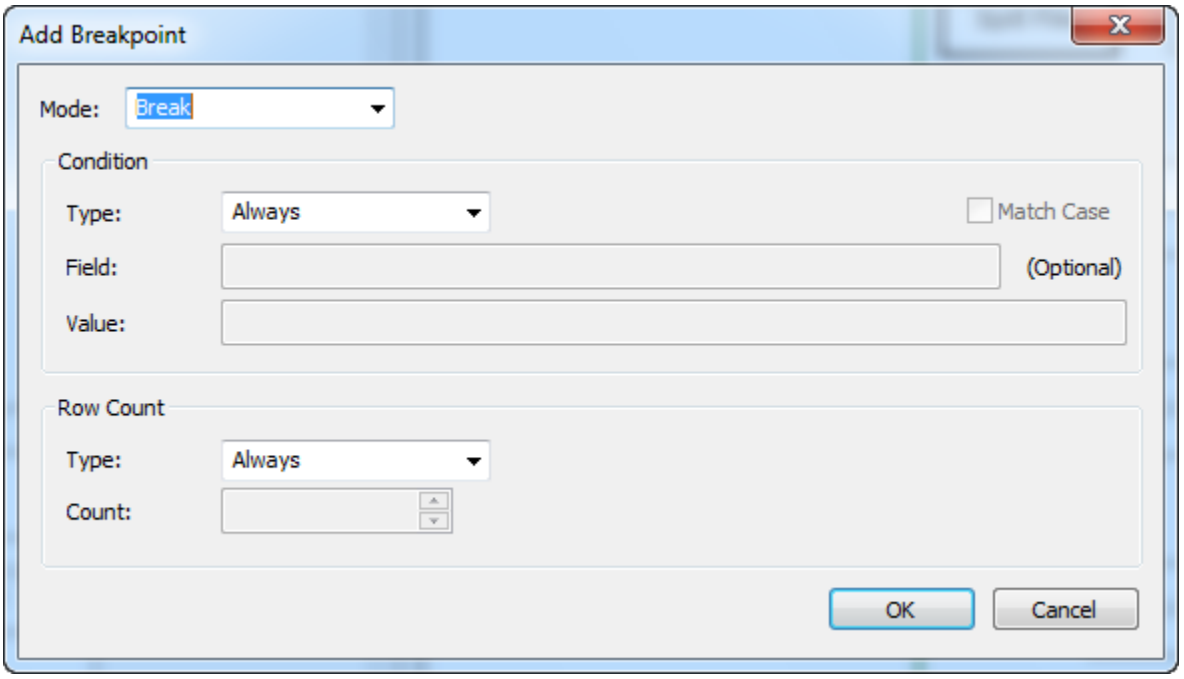
- Continue pressing Edge/Step watching the debugger go through each row of data until it reaches the end of that graph.
- Press Edge/Step again to see a new graph start.
- Notice that contents of the Debug Information window change as the debugging process passes over each edge.
- Click on an edge or vertex and view the related details in the Properties window.
- Once the debugging process has completed, the results of your query are displayed.

Navigating the graph using ECL

- Click on Graph/Step Start. The next active graph is loaded.
- In the Debug Information window, select the Builder tab.
- Scroll down to the line of ECL which defines what you want to locate.
- Left click on the red circle and select the number of the vertex to be located.
- Selecting the edge of the selected vertex, right click and use the Continue to.. option. The debugging process will continue until the first row of data arrives at the edge you selected.
- Click on Edge/Next. The debugging process continues until a new row arrives at the active edge.

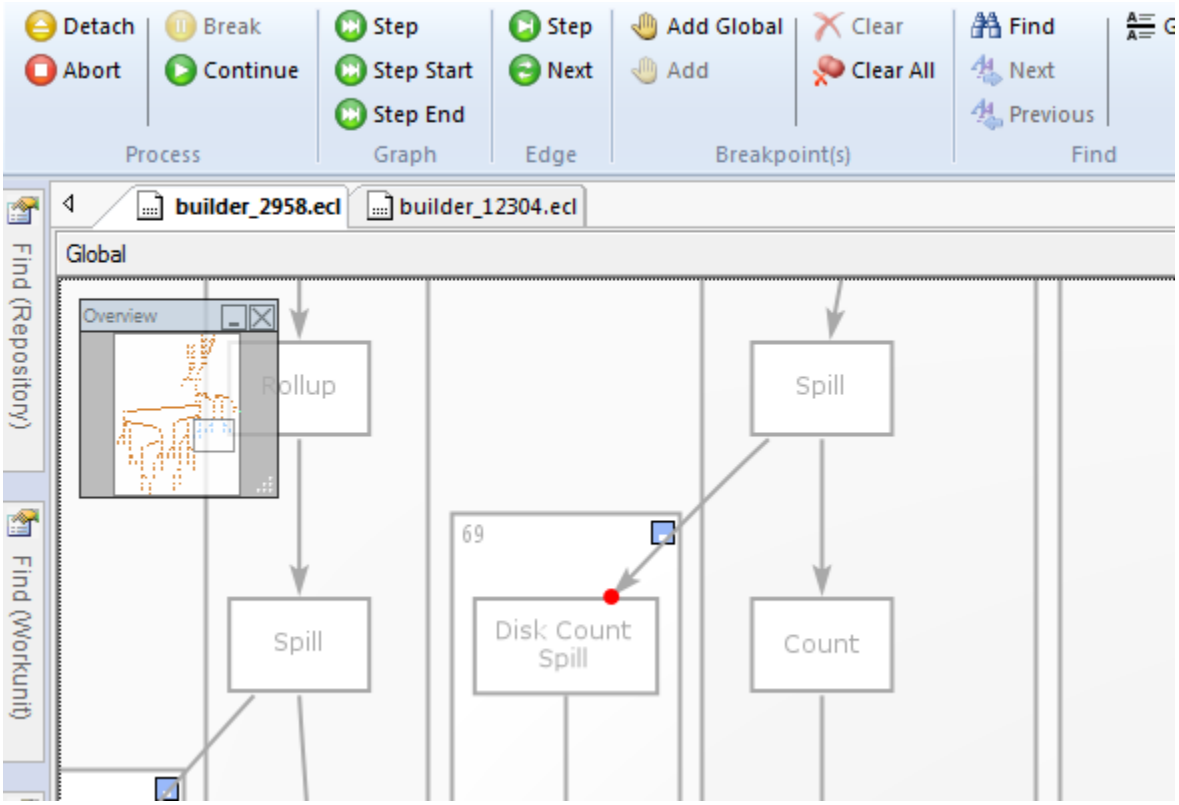
Setting global breakpoints

- Click on Breakpoint(s)/Add Global.
- Select the Mode required.
- Set the Conditions to be applied.
- Click the OK button.



Setting local breakpoints

- Select the edge on the graph where you want the breakpoint to be set.
- Click on Breakpoint(s)/ Add, or right click and select Add Breakpoint.
- Select the Mode required.
- Set the conditions to be applied.
- Click the OK button. The breakpoint is shown on the graph as a red dot.



Note: Simple breakpoints that do not require any conditions to be set, are quickly added by selecting an edge and using the right-click option Set Breakpoint. Breakpoints set in this way use the default settings which are the Mode setting Break and the Condition Always.

Setting a breakpoint on a line of ECL

- Using the Debug Information window, click on the Builder tab.
- Scroll the any line which begins with a red circle.
- Left click on the red circle and select Set Breakpoint. Two breakpoints are added to the graph.

Removing breakpoints

To remove a single breakpoint:

- Highlight the edge where the breakpoint is shown and click on Breakpoints/Clear, or right-click and select Clear Breakpoint(s).
- Single breakpoints can also be removed by right-clicking on the breakpoint listed in the Breakpoints window and selecting delete.

To remove all breakpoints on the graph:

- Click on Breakpoint(s)/Clear All.

Using the search window

- Enter your search criteria in Field and Value.
- Click on the Search button. Matches for your search are located and the Edge IDs are displayed.
- Click on the Edge IDs to locate each match on the graph.
- View the Debug Information window to see the data content which shows your search criteria highlighted.

Note: Only the last 10 rows of data that flowed through any given edge are searched.

ECL Plus

Command Line Interface

eclplus.exe

eclplus *action= owner= user= password= cluster= server= queue= graph= timeout= ecl= file= format= output= jobname= -debugparam1= _applicationparam1= /variablename1=*

action=	One of the following options: list|view|dump|delete|abort|query|graph(the default option is “query”).
owner=	The workunit owner.
user=	The userid.
password=	The password authorizing access for the user.
cluster=	The name of the cluster to use.
server=	The IP address or DNS name of the ECL Watch server.
queue=	The name of the job queue.
graph=	The name of graph.
timeout=	Query timeout in seconds (0 for asynchronous).
ecl=	The ECL code to execute. Optionally, this may be replaced by the name of an input file containing the ECL to execute (in the form: @inputfile).
file=	The logical name of the file, or the logical name with the starting and ending rows specified (in the form: !logicalName[startrow,endrow]).
format=	One of the following options: default | csv | csvh | xml | runecl | bin(ary)
output=	The name of the file to output.
jobname=	The name to give the job.
pagesize=	The number of rows per page. If omitted, the default is 500.
-debugparam=	Debug parameters to pass on the command line, in the form: -debugparam=debugvalue
_applicationparam=	Parameters to pass on the command line, in the form: _applicationparam=applicationvalue
/variablename=	Variables to pass on the command line, in the form: /variablename=[(int)|(bool)] valueThe default value type is string unless int or bool is specified (in parentheses preceding the value). The *variablename* is the STORED name of an EXL file in your ECL code.

The **eclplus** executable accepts command line parameters to send directly to an ECL execution engine. These options can be typed directly on the command line, sent using a script or batch file, through an **ini** file in the same directory as the executable, or any combination.

eclplus.ini

All the options can be put directly on the command line, or placed in a file called eclplus.ini in the same directory as the executable. If your operating system is case-sensitive, make sure the filename is in lowercase. Options that do not change very often can be put in the ini file. For example:

```
server=10.150.50.12
cluster=training
queue=trainingQueue
user=rtor
password=password
```

In all the examples below, we'll assume eclplus.ini has the above content.

We do not recommend storing your password in the ini file (which is clear text). The password is included in the ini file for these examples to simplify the example code.

Running queries in batch mode

Batch mode queries are executed using the *ecl=* option, in any of its three forms. In the first form you simply put your ECL code on the command line itself:

```
eclplus ecl=1+1
            // Result = 2
```

In the second form, your ECL code is in an input file. For example, assume you have a text file called dataset.txt, which contains the following ECL code:

```
myrec := record
string10 firstname,
string10 lastname
end;
ds := dataset([{'Yanrui', 'Ma'}, {'Richard', 'Taylor'},
{'Richard', 'Chapman'}], myrec);
output(ds, ,'testdata::namesdb');
```

Then if you run:

```
eclplus @dataset.txt
```

A dataset will be created and the result will be written to the thor file testdata::namesdb.

If also have a text file called datasetquery.txt containing:

```
myrec := record
string10 firstname,
string10 lastname
end;
ds1 := dataset('testdata::namesdb', myrec, thor);
output(ds1);
```

then run:

```
eclplus @datasetquery.txt
```

You'll get:

```
firstname lastname
Yanrui Ma
Richard Taylor
Richard Chapman
```

Workunit manipulation

A workunit is a data structure that is passed among eclplus, daliserver, and eclccserver. It contains real-time information about the query, so you can control the process of a query by manipulating the workunit.

List all work units

To list all work units:

```
eclplus action=list
```

The output looks like:

```
WUID OWNER JOBNAME STATUS
W20090226-100258-85132143 yma dataset.txt completed
W20090226-100958-85552898 yma datasetquery.txt completed
```

Each workunit has a WUID (WorkUnit IDentifier), owner, jobname and status. You can see that the jobname is simply the filename that contains the query, but you can specify the jobname by your self, like this:

```
eclplus jobname=myquery1 @datasetquery.txt
```

View the result of a certain workunit

You can look at specific workunit results, like this:

```
eclplus action=view wuid=
          W20090226-100958-85552898
```

The output will look like:

```
firstname lastname
Yanrui Ma
Richard Taylor
Richard Chapman
```

Dump a workunit

If you want to get all the details describing a workunit, use the dump option for the action parameter:

```
eclplus action=dump wuid= W20090226-100958-85552898
```

See the Workunit Dump section below for the result.

Return graph data for a workunit:

This action returns the XML data for one or more workunit graphs.

```
eclplus action=graph graph=graph1 wuid=W20090226-100958-85552898
```

Graph name must be supplied in the graph= parameter.

Aborting a workunit

If a query is taking an usually long time and you doubt something is wrong, you can abort it by:

```
eclplus action=abort wuid= W20090226-100958-85552898
```

You can use list to find out the wuid the workunit and use abort to abort it.

Timeout

Before you run a query, if you know the query is going to take a long time, you can specify a timeout, then your eclplus will return when it reaches the timeout, and the query will run in the background.

For example:

```
eclplus @datasetquery.txt timeout=0
```

eclplus will return immediately.

```
eclplus @datasetquery.txt timeout=2
```

eclplus will return in 2 seconds.

You can list/view the workunit associated with the query to monitor its status.

Output format

By default, the result displays on the screen. You can direct it to a file, by using the output option:

```
eclplus @datasetquery.txt output=o1.txt
cat o1.txt
firstname lastname
Yanrui Ma
Richard Taylor
Richard Chapman
```

Also, you may specify the following output formats:

csv

```
eclplus @datasetquery.txt format=csv
[QUERY 0]
"Yanrui ","Ma "
"Richard ","Taylor "
"Richard ","Chapman "
```

csvh

```
eclplus @datasetquery.txt format=csvh
[QUERY 0]
"firstname","lastname"
"Yanrui ","Ma "
"Richard ","Taylor "
"Richard ","Chapman "
```

raw

```
eclplus @datasetquery.txt format=raw
Yanrui     Ma
Richard    Taylor
Richard    Chapman
```

runecl

```
eclplus @datasetquery.txt format=runecl
[QUERY 0]
```

```
[0]
firstname -> Yanrui
lastname -> Ma
[1]
firstname -> Richard
lastname -> Taylor
[2]
firstname -> Richard
lastname -> Chapman
```

bin(ary)

```
eclplus @datasetquery.txt format=bin
Yanrui Ma Richard Taylor Richard Chapman
```

Workunit Dump

A Workunit dump is an XML representation of every piece of data in the workunit. This contains all the information that you could discover about the workunit by using ECL Watch.

The following workunit dump came from a simple COUNT(person) query in the Training environment:

```
<W20110615-160604 agentPID="4162" agentSession="4296042782" cloneable="1"
 clusterName="thor" codeVersion="138"  isClone="1" scope="hpccdemo"
 state="completed" submitID="hpccdemo"
 token="X1lUMJ6oacON/1anTHTQW1JVHr1bbY8EWTSJhlDOrtYxmD13Z5ly4Qd26sEYVtxhW">
  <Action>run</Action>
  <Debug>
    <applyinstantecltransformations>1</applyinstantecltransformations>
    <applyinstantecltransformationslimit>100</applyinstantecltransformationslimit>
    <created_by>ws_workunits</created_by>
    <created_for>hpccdemo</created_for>
    <eclagentlog>//192.168.237.132/var/log/HPCCSystems/myeclagent/eclagent.06_15_11.log
    </eclagentlog>
    <targetclustertype>hthor</targetclustertype>
  </Debug>
  <Query fetchEntire="1">
    <Associated>
      <File crc="701142319" filename="libW20110615-160604.so" type="dll"/>
    </Associated>
    <Text>
      <Archive build="community_3.0.0" eclVersion="3.0.0">  <Query
       originalFilename="C:\DOCUME~1\Hpccdemo\LOCALS~1\Temp\TFR2CE.tmp">
       OUTPUT(&amp;apos;Hello World&amp;apos;); </Query> </Archive>
    </Text>
  </Query>
  <resultLimit>100</resultLimit>
  <Results>
    <Result fetchEntire="1" name="Result 1" sequence="0" status="calculated">
      <rowCount>1</rowCount>
      <SchemaRaw xsi:type="SOAP-ENC:base64"> UmVzdWx0XzEABPH///8BYXNjaWkAAWFzY2lpAAAYAAAAAA==
      </SchemaRaw>
      <totalRowCount>1</totalRowCount>
      <Value xsi:type="SOAP-ENC:base64"> CwAAAEhlbGxvIFdvcmxk </Value>
    </Result>
  </Results>
  <TimeStamps>
    <TimeStamp application="workunit">
      <Created ts="1308153964"> 2011-06-15T16:06:04Z </Created>
    </TimeStamp>
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Started ts="1308153971"> 2011-06-15T16:06:11Z </Started>
    </TimeStamp>
```

```
    <TimeStamp application="EclAgent" instance="localhost.localdom">
      <Finished ts="1308153971"> 2011-06-15T16:06:11Z </Finished>
    </TimeStamp>
  </TimeStamps>
  <Timings>
    <Timing count="1" duration="1" max="1308040" name="WorkUnit_lockRemote"/>
    <Timing count="1" duration="6" max="6577412" name="SDS_Initialize"/>
    <Timing count="1" duration="0" max="704338" name="Environment_Initialize"/>
    <Timing count="1" duration="16" max="16414003" name="Process"/>
  </Timings>
  <Workflow>
    <Item mode="normal" state="done" type="normal" wfid="1">
      <Schedule/>
    </Item>
  </Workflow>
</W20110615-160604>
```

ECL Command Line Interface

The ECL Command Syntax

ecl [--version] <command> [<options>]

--version	displays version info.
Arguments	
deploy	Create a workunit from an ecl file, archive, or dll
publish	Add a workunit to a query set
unpublish	Remove a query from a query set
run	Run the given ecl file, archive, dll, wuid, or query
activate	Activate a published query
deactivate	Deactivate the given query alias name
queries	List or manipulate queries and querysets
roxie	execute commands for Roxie
packagemap	execute packagemap commands (for Roxie)
bundle	manage ECL bundles
abort	aborts one or more workunits from the given WUID or job name
status	returns the status of a given workunit or job name. If more than one is found, a list returns.
getname	returns the workunit name from the given WUID.
getwuid	returns the WUID(s) of the given workunit job name.

ecl.ini

Many options can be placed in a file called **ecl.ini** in the same directory as the executable. Options that do not change very often should be put in the ini file. For example:

```
eclWatchIP=10.150.50.12
eclWatchPort=28010
eclUserName=emilykate
eclPassword=elmo812
resultLimit=200
```

In some examples below, we'll assume ecl.ini has the above content.

We do not recommend storing your password in the INI file (which is clear text). The password is included in the INI file for these examples to simplify the example code.

The following options can be provided in an ini file: eclWatchIP, eclWatchPort, eclUserName, eclPassword, activateDefault, waitTimeout, resultLimit.

Evaluation of options follows this order of precedence:

- command line
- ini file
- environment variable
- default value

Environment Variables

Some options can be stored in Environment Variables on your machine. The following options are supported:

```
ECL_WATCH_IP
ECL_WATCH_PORT
ECL_USER_NAME
ECL_PASSWORD
ECL_WAIT_TIMEOUT
ECL_RESULT_LIMIT
```

We do not recommend storing your password in an Environment Variable unless your system is secured.

ecl deploy

ecl deploy --target=<value> --name=<value> <ecl_file | - >

ecl deploy --target=<value> --name=<value> <archive | - >

ecl deploy [--target=<value>] [--name=<value>] <so | dll | - >

Examples:

```
ecl deploy --target=roxie --name=FindPersonService findperson.ecl
ecl deploy --target=roxie --name=FindPersonService ArchiveQuery.xml
ecl deploy --target=roxie --name=FindPersonService libW20120224-125557.so
```

A hyphen (-) specifies that the object should be read from stdin.

ecl deploy	Creates a workunit on the HPCC system from the given ECL text, file, archive, shared object, or dll. The workunit is created in the *compiled* state.
Arguments	
ecl_file	The ECL text file to deploy
archive	TheECL archive to deploy
so | dll	The workunit dynamic linked library or shared object to deploy
Options	
-t, --target	The target cluster to associate workunit with
-n, --name	The workunit query name
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--main	The definition to use from legacy ECL repository
--ecl-only	Send ecl text to hpcc without generating archive
eclcc Options	
-Ipath	Add path to locations to search for ecl imports
-Lpath	Add path to locations to search for system libraries
--manifest	Specify path to manifest file

ecl publish

ecl publish [--target=<val>] [--name=<val>] [--activate | --no-activate] <wuid>

ecl publish [--target=<val>] [--name=<val>] [--activate | --no-activate] <so | dll | - >

ecl publish --target=<val> --name=<val> [--activate | --no-activate] <archive | - >

ecl publish --target=<val> --name=<val> [--activate | --no-activate] <ecl_file | - >

Examples:

```
ecl publish --target=roxie --name=FindPersonService -A W20120224-125557
ecl publish --target=roxie --name=FindPersonService -A libW20120224-125557.so
ecl publish --target=roxie --name=FindPersonService -A ArchiveQuery.xml
ecl publish --target=roxie --name=FindPersonService --activate findperson.ecl
ecl publish --target=roxie --name=FindPersonService --activate --no-files findperson.ecl
ecl publish --target=roxie --name=FindPersonService --no-activate findperson.ecl
```

A hyphen (-) specifies that the object should be read from stdin.

ecl publish	Publishes a query into a queryset. The query is created by adding a workunit to a queryset and assigning it a query name.
Arguments	
wuid	The workunit id to publish
ecl_file	The ECL text file to deploy
archive	The ECL archive to deploy
so | dll	The workunit dynamic linked library or shared object to deploy
Options	
-t, --target	The target cluster to associate workunit with
-n, --name	The workunit job name
-A, --activate	Activates query when published (default)
-A-, --no-activate	Does not activate query when published
-sp, --suspend-prev	Suspend previously active query
-dp, --delete-prev	Delete previously active query
--no-files	Specifies to not copy DFS file information for files referenced by the query
--no-reload	Specifies to not request a reload of the Roxie cluster
--allow-foreign	Specifies to allow the use of foreign files in a Roxie query. If a Roxie query references foreign files and this is not enabled, publish will fail.
--timeLimit=<sec>	Value to set for query timeLimit configuration
--warnTimeLimit=<sec>	Value to set for query warnTimeLimit configuration
--memoryLimit=<mem>	Value to set for query memoryLimit configuration. Format <mem> as 500000B, 550K, 100M, 10G, or 1T, etc.
--wait=<sec>	Maximum time to wait for cluster finish updating
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.

-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--main	The definition to use from legacy ECL repository
--ecl-only	Send ecl text to hpcc without generating archive
eclcc Options	
-Ipath	Add path to locations to search for ecl imports
-Lpath	Add path to locations to search for system libraries
--manifest	Specify path to manifest file

ecl unpublish

ecl unpublish <queryset> <query_id>

Example:

```
ecl unpublish roxie FindpersonService.1
ecl unpublish roxie "FindpersonService*"
```

ecl unpublish	executes the supplied ecl unpublish command
Arguments	
queryset	The name of queryset containing query to unpublish
query_id	The query to remove from query set. Wildcards allowed, but must be in quotes (e.g., "MyQuery*").
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl run

ecl run [--target=<val>][--input=<file|xml>][--wait=<ms>] <wuid>

ecl run [--target=<c>][--input=<file|xml>][--wait=<ms>] <queryset> <query>

ecl run [--target=<c>][--name=<nm>][--input=<file|xml>][--wait=<i>] <dll|->

ecl run --target=<c> --name=<nm> [--input=<file|xml>][--wait=<i>] <archive|->

ecl run --target=<c> --name=<nm> [--input=<file|xml>][--wait=<i>] <eclfile|->

Examples:

```
ecl run --target=thor --input=data.xml --wait=1000 W20120224-125557
ecl run --target=thor --input=data.xml --wait=1000 thor findpersonservice
ecl run --target=thor --input=data.xml --wait=1000 ArchiveQuery.xml
ecl run --target=thor --input=data.xml --wait=1000 findperson.ecl
ecl run --target=thor --input="<request><LName>JONES</LName></request>" findperson.ecl
```

A hyphen (-) specifies that the object should be read from stdin.

ecl run	executes the supplied ecl run command
Arguments	
wuid	The workunit id to publish
ecl_file	The ECL text file to deploy
archive	TheECL archive to deploy
so | dll	The workunit dynamic linked library or shared object to deploy
Options	
-t, --target	The target cluster to associate workunit with
-n, --name	The workunit job name
-in,--input=<file|xml>	The file or xml content to use as query input
-X<name>	Sets the stored input value (stored('name'))
--wait=<sec>	Maximum time to wait for cluster finish updating (in ms)
--exception-level	Sets the minimum severity for reporting exceptions. Possible severity levels are **info**, **warning**, or **error**. The default is info which returns all exceptions.
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--main	The definition to use from legacy ECL repository
--ecl-only	Send ecl text to hpcc without generating archive
--limit	Sets the result limit for the query, defaults to 100
-f<option>[=value]	set an ECL option (equivalent to #OPTION in ECL)
eclcc Options	
-Ipath	Add path to locations to search for ecl imports

-Lpath	Add path to locations to search for system libraries
--manifest	Specify path to manifest file

ecl activate

ecl activate <queryset> <query_id>

Example:

```
ecl activate Roxie FindpersonService.4
```

ecl activate	Activates a published query. This assigns a query to the active alias with the same name as the query.
Arguments	
queryset	The name of queryset containing query to activate
query_id	The query to activate
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl deactivate

ecl deactivate <queryset> <active_alias>

Example:

```
ecl deactivate Roxie FindpersonService
```

ecl deactivate	Deactivates a published query by removing an active query alias from the given queryset.
Arguments	
queryset	The name of queryset containing alias to deactivate
active_alias	The active alias to be removed from the queryset
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl queries list

ecl queries list [<queryset>][--target=<cluster>][--show=<flags>]

Examples:

```
ecl queries list roxie
ecl queries list roxie --target=roxie --show=A
```

ecl queries list	Displays a list of the queries in one or more querysets. If a cluster is provided the querysets associated with that cluster will be shown. If no queryset or cluster is specified all querysets are shown.
Actions	
list	List queries in queryset(s)
Options	
queryset	The name of queryset from which to list queries
-t, --target	The target cluster associated with the queries to list
-A, --activate	Activates query when published
--show=<flags>	Show only queries with matching flags
Flags	
A	Active
S	Suspended
U	No Flags
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl queries copy

ecl queries copy <source_query_path> <target_queryset> [--activate]

Examples:

```
ecl queries copy thor/findperson thor2 --activate
ecl queries copy //192.168.1.10:8010/thor/findperson thor
```

ecl queries copy	Copies a query from one queryset to another. A query can be copied from one HPCC environment to another by using a path which begins with '//' followed by the IP or hostname and Port of the source ECL Watch and then followed by the source queryset and query.
Actions	
copy	Copy a query from one queryset to another
Options	
source_query_path	The path of the query to copy using the format: [//ip:port/]queryset/query or queryset/query.
target_queryset	The name of the queryset to which the query should be copied
-t, --target	The target cluster to associate with the remote workunit
--no-files	Specifies to not copy DFS file information for files referenced by the query
-A, --activate	Activates query when copied
--no-reload	Specifies to not request a reload of the Roxie cluster
--allow-foreign	Specifies to allow the use of foreign files in a Roxie query. If a Roxie query references foreign files and this is not enabled, copy will fail.
-O, --overwrite	Whether to overwrite existing information - true if present
--timeLimit=<sec>	Value to set for query timeLimit configuration
--warnTimeLimit=<sec>	Value to set for query warnTimeLimit configuration
--memoryLimit=<mem>	Value to set for query memoryLimit configuration. Format <mem> as 500000B, 550K, 100M, 10G, or 1T, etc.
--wait=<sec>	Maximum time to wait for cluster finish updating (in ms)
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl queries copy-set

ecl queries copy-set <source_target> <destination_target> [--all] [--clone-active-state]

Examples:

```
ecl queries copy-set roxie1 roxie2
ecl queries copy-set roxie1 roxie2 --all
ecl queries copy-set roxie1 roxie2 --clone-active-state
```

ecl queries copy-set	Copies a set of queries from one target to another.
Actions	
copy-set	Copy a set of queries from one target to another.
Options	
source_target	Target cluster from which to copy queries.
destination_target	Target cluster to copy queries to.
--all	Specifies to copy both active and inactive queries. If omitted, only active are copied.
--no-files	Specifies to not copy DFS file information for files referenced by the query
--daliip=	IP address or hostname of the remote Dali to use for logical file lookups.
--source-process	Process cluster from which to copy files.
--clone-active-state	Make copied queries active on target if they are active on the source.
--allow-foreign	Specifies to allow the use of foreign files in a Roxie query. If a Roxie query references foreign files and this is not enabled, copy will fail.
-O, --overwrite	Whether to overwrite existing DFS information - true if present
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl, --ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl queries config

ecl queries config <target> <queryid> [options]

Examples:

```
ecl queries config thor findperson --wait=1000
```

ecl queries config	Updates query configuration values
Actions	
config	Set or update query configuration values
Options	
target	The name of the target queryset
queryid	The name of the query
--no-reload	Specifies to not request a reload of the Roxie cluster
--wait=<sec>	Maximum time to wait for cluster finish updating (in ms)
--timeLimit=<sec>	Value to set for query timeLimit configuration
--warnTimeLimit=<sec>	Value to set for query warnTimeLimit configuration
--memoryLimit=<mem>	Value to set for query memoryLimit configuration. Format <mem> as 500000B, 550K, 100M, 10G, or 1T, etc.
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap add

ecl packagemap add [--daliip][options] <target> <filename>

Examples:

```
ecl packagemap add  -s=192.168.1.10 roxie mypackagemap.pkg
ecl packagemap add roxie mypackagemap.pkg --overwrite
ecl packagemap add roxie mypackagemap.pkg --daliip=192.168.11.11
```

ecl packagemap add	Calls the packagemap add command
Actions	
add	Adds a packagemap to the target cluster
Arguments	
target	The target to associate the packagemap with
filename	The name of the file containing packagemap information.
--daliip=	IP address or hostname of the remote Dali to use for logical file lookups
Options	
-O, --overwrite	Whether to overwrite existing information - true if present
-A, --activate	Activates packagemap
--allow-foreign	Specifies to allow the use of foreign files. If a packagemap references foreign files and this is not enabled, packagemap add will fail.
--pmid=<packagemapid>	id of package map - defaults to filename if not specified
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap delete

ecl packagemap delete [options] <target><packagemap>

Examples:

```
ecl packagemap delete roxie mypackagemap
```

ecl packagemap delete	Calls the packagemap delete command
Actions	
delete	Deletes a packagemap
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap activate

ecl packagemap activate <target> <packagemap>

Example:

```
ecl packagemap activate roxie mypackagemap.pkg
```

ecl packagemap activate	The activate command will deactivate the currently active packagemap and make the spec packagemap active.
Arguments	
target	The target containing the packagemap to activate
packagemap	name of packagemap to update
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap deactivate

ecl packagemap deactivate <target> <packagemap>

Example:

```
ecl packagemap deactivate roxie mypackagemap.pkg
```

ecl packagemap deactivate	The deactivate command will deactivate the currently active packagemap.
Arguments	
target	The target containing the packagemap to deactivate
packagemap	Name of packagemap to deactivate
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap list

ecl packagemap list <target>

Examples:

```
ecl packagemap list roxie
```

ecl packagemap list	Calls the packagemap list command
Actions	
list	Lists loaded packagemap names
Arguments	
target	The target containing the packagemap to list
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap info

ecl packagemap info [options] <target>

Examples:

```
ecl packagemap info roxie
```

ecl packagemap info	Calls the packagemap info command
Actions	
info	returns packagemap info
Arguments	
target	The target containing the packagemap to retrieve
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl packagemap validate

ecl packagemap validate <target> [<filename>]

Examples:

```
ecl packagemap validate roxie mypackagemap.pkg
ecl packagemap validate roxie --active
```

The packagemap validate command verifies that :

- Referenced superkeys have subfiles defined (warns if no subfiles exist)
- All referenced queries exist in the current Roxie queryset
- All Roxie queries are defined in the package

The result will also list any files that are used by queries but not mapped in the packagemap.

Filename, --active, and --pmid are mutually exclusive. The --active or --pmid options validate a packagemap that has already been added instead of a local file.

The --queryid option checks the files in a query instead of all the queries in the target queryset. This is quicker when you only need to validate the files for a single query.

ecl packagemap validate	Calls the packagemap validate command.
Actions	
validate	Validates packagemap info
Arguments	
filename	The filename containing the packagemap info to validate
target	The target containing the packagemap to validate
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--active	Validates the packagemap that is active for the given target
--pmid=<packagemapid>	Validates the given packagemap
--queryid	Validate the files for the given queryid if they are mapped in the packagemap
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl roxie attach

ecl roxie attach <processName>

Examples:

```
ecl roxie attach myroxie
```

ecl roxie attach	Attach the roxie to Dali
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl roxie detach

ecl roxie detach <processName>

Examples:

```
ecl roxie detach myroxie
```

ecl roxie detach	Detach the roxie from Dali
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl roxie reload

ecl roxie reload <processName>

Examples:

```
ecl roxie reload myroxie
```

ecl roxie reload	Reloads the roxie info from Dali
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl roxie check

ecl roxie check <processName>

Examples:

```
ecl roxie check myroxie
```

ecl roxie check	Checks the state of the roxie process
Options	
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-ssl	Use SSL to secure the connection to the server.
--wait=<*ms*>	Max time to wait in milliseconds
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl bundle depends

ecl bundle depends <bundleName> [--version <versionnumber>]

Examples:

```
ecl bundle depends mybundle
ecl bundle depends mybundle --version=2
```

ecl bundle depends	Shows the dependencies of a bundle
Options	
<bundleName>	The name of a bundle file or installed bundle
--recurse	Displays indirect dependencies
--version	Specify a version of the bundle
-v, --verbose	Output additional tracing information

ecl bundle info

ecl bundle info <bundleName> [--version <versionnumber>]

Examples:

```
ecl bundle info mybundle
ecl bundle info mybundle --version=2
```

ecl bundle info	Lists information about a bundle
Options	
<bundleName>	A bundle filename, a bundle folder, or a bundle name.
--version	Specify a version of the bundle
-v, --verbose	Output additional tracing information

ecl bundle install

ecl bundle install <bundleName>

Examples:

```
ecl bundle install mybundle
ecl bundle install mybundle --dryrun
ecl bundle install mybundle --update
ecl bundle install mybundle --keepprior
```

ecl bundle install	Installs a bundle
Options	
<bundleName>	A bundle filename or a bundle folder.
--dryrun	List what would be installed, but do not copy
--force	Install even if required dependencies missing
--keepprior	Do not remove any previous versions of the bundle
--update	Update an existing installed bundle
-v, --verbose	Output additional tracing information

ecl bundle uninstall

ecl bundle uninstall <bundleName>

Examples:

```
ecl bundle uninstall mybundle
ecl bundle install mybundle --dryrun
ecl bundle install mybundle --update
ecl bundle install mybundle --keepprior
```

ecl bundle install	Installs a bundle
Options	
<bundleName>	The name of an installed bundle
--dryrun	List what would be removed, but do not remove them
--force	Uninstall even if other bundles are dependent on this
--version	Specify a version of the bundle
-v, --verbose	Output additional tracing information

ecl bundle list

ecl bundle list <pattern>

Examples:

```
ecl bundle list
ecl bundle list myb*
```

ecl bundle list	Lists bundles matching specified pattern
Options	
<pattern>	A pattern specifying bundles to list. If omitted, all bundles are listed
--details	Report details of each installed bundle
-v, --verbose	Output additional tracing information

ecl bundle use

ecl bundle use <bundleName> [--version <version>]

Example:

```
ecl bundle use myBundle --version 2
```

ecl bundle use	Makes a specified version of a bundle active
Options	
<bundleName>	The name of a bundle file
--version	The version of the bundle to make active, or "none"
-v, --verbose	Output additional tracing information

ecl roxie unused-files

ecl roxie unused-files <processName>

Examples:

```
ecl roxie unused-files myroxie
```

ecl roxie unused-files	Finds files in the DFS for the given roxie process that are not currently used by queries on that roxie.
Options	
--check-packagemaps	Exclude files referenced in active packagemaps
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl abort

ecl abort -wu <WUID> | -n <jobName>

Examples:

```
ecl abort -wu W20150516-111213
ecl abort -n MyJob
```

ecl abort	aborts one or more Workunits from the given WUID or job name
Options	
-wu	The WUID (Workunit ID)
-n	The job name
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl,--ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl status

ecl status -wu <WUID> | -n <jobName>

Examples:

```
ecl status -wu W20150516-111213
ecl status -n MyJob
```

ecl status	returns the status of a given workunit or job name. If more than one is found, a CSV list returns.
Options	
-wu	The WUID (Workunit ID)
-n	The job name
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl,--ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl getwuid

ecl getwuid -n <jobName> [--limit=<limitCount>]

Examples:

```
ecl getwuid -n MyJobName
ecl getwuid -n MyCommonJobName --limit=100
```

ecl getwuid	returns the WUID(s) for a given job name. If more than one is found, a list returns.
Options	
-n	The job name
--limit=*nn*	Integer to set result limit, default is 100
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl,--ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ecl getname

ecl getname -wu <WUID>

Examples:

```
ecl getname -wu W20140516-111213
ecl getname -wu W201407*
```

ecl getname	returns the job name for a given workunit.
--wuid	The WUID (Workunit ID)
Options	
--limit=<limit>	This sets the result limit. This is useful when using wildcards in a request. (Default is 100)
-v, --verbose	Output additional tracing information
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
-ssl,--ssl	Use SSL to secure the connection to the server.
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)

ECL Compiler

The ECL Compiler is the compiler component of the High Performance Computing Cluster (HPCC). It is embedded and included when you install the HPCC. The compiler is the component that actually compiles the ECL code.

The syntax and many of the compiler options implemented are similar to the gcc compiler. You can execute either the Linux or Windows version of eclcc, which, when run, load several of our shared objects (SO files, on Linux) or DLLs (on Windows). The ECL Compiler can process hThor, Thor, or Roxie targeted ECL code.

To compile and run ECL code locally on your Windows machine, you will need the Microsoft Visual Studio 2008 C++ compiler (either Express or Professional edition). This is available from http://www.microsoft.com/express/Downloads/#2008-Visual-CPP

Using the ECL Compiler as a Stand Alone option

The ECL Compiler is normally used through the ECL IDE or Eclipse using the ECL plug-in for Eclipse, however, you can use the ECL Compiler in a stand alone manner, to create stand alone programs, or workunits. The ECL Compiler can read ECL code from standard input, or can read it from a specified input file. It compiles the code into an executable program (Such as an 'EXE' file in Windows). The resulting program, when executed, runs the job, writing any output to standard output. Alternatively, you could redirect the output to a file or pipe into another process. With the ECL Compiler, you do not need a supercomputer cluster to develop and run ECL code.

Running the ECL Compiler without any options (or specifying –help) will display the syntax.

```
C:\eclcc>eclcc -help
```

Usage: eclcc <options> ECL_file.ecl

General options:

-I <*path*>	Add path to locations to search for ecl imports
-L <*path*>	Add path to locations to search for system libraries
-o <*file*>	Specify name of output file (default a.out if linking to executable, or stdout)
-manifest	Specify path to manifest file listing resources to add
-foption[=value]	Set an ecl option. See #OPTION in the *ECL Language Reference* for details.
-main <*ref*>	Compile definition <ref> from the source collection
-syntax	Perform a syntax check of the ECL
-platform=hthor	Generate code for hthor executable (default)
-platform=roxie	Generate code for roxie cluster
-platform=thor	Generate code for thor cluster

NOTE: If there are spaces in the path you specify, put it in quotes. For example: –L"C:\Program Files"

Output control options:

-E	Output preprocessed ECL in xml archive form
-M	Output meta information for the ecl files
-Md	Output dependency information
-Me	eclcc should evaluate supplied ecl code rather than generating a workunit
-q	Save ECL query text as part of workunit
-wu	Only generate workunit information as xml file

C++ options:

-S	Generate c++ output, but don't compile
-c	Compile only (don't link)
-g	Enable debug symbols in generated code
-Wc,xx	Pass option xx to the c++ compiler
-D*name=value*	Override the definition of a global attribute 'name'
-Wl,xx	Pass option xx to the linker
-Wa,xx	Pass straight through to c++ compiler
-Wp,xx	Pass straight through to c++ compiler
-save-cpps	Do not delete generated c++ files (implied if -g)
-shared	Generate workunit shared object instead of a stand-alone executable

Other options:

--allow=str	Allow use of named feature. (e.g., cpp, pipe, all)
	cpp: Allow embedded code within ECL (e.g., c++, JAVA, Javascript, Python, R, etc.)
	pipe: Allow the PIPE command to send data to an external program.
	all: Allow all features
-b	Batch mode. Each source file is processed in turn. Output name depends on the input filename
-checkVersion	Enable/disable ecl version checking from archives
--deny=all	Disallow use of all named features not specifically allowed using --allow
--deny=str	Disallow use of named feature
	cpp: Disallow embedded code within ECL (e.g., c++, JAVA, Javascript, Python, R, etc.)
	pipe: Disallow the PIPE command to send data to an external program.
-help, --help	Display help message
--help -v	Display verbose help message
--internal	Run internal tests
--legacy	Use legacy import semantics (deprecated)
--keywords	Outputs the lists of ECL reserved words to stdout (XML format)
--logfile <*file*>	Write log to specified file
--logdetail=*n*	Set the level of detail in the log file
-specs <*file*>	Read eclcc configuration from specified file
-split *m:n*	Process a subset m of n input files (only with -b option)
-v --verbose	Output additional tracing information while compiling
--version	Output version information
--timings	Output additional timing information

Compiled Options:

After you have successfully compiled the code, it produces an executable file. There are a few additional options that can be used when running that executable.

Usage: a.out <options>

-wu=<file>	Write XML formatted workunit to given filespec and exit
-xml	Display output as XML
-raw	Display output as binary
-limit=x	Limit number of output rows
--help	Display help text

Examples

The following example demonstrates what you can do once the ECL Compiler is installed and operational.

Running a basic ECL program using the command line compiler

Once the ECL Compiler is installed, you can use the ECL Compiler to run an ECL program.

- Create a file called hello.ecl, and type in the text

  ```
  Output('Hello world');
  ```

  (including the quotes) into the file.

  You can either use your favorite editor, or you can use the command line by typing the following (for Windows systems):

  ```
  echo Output('Hello world'); > hello.ecl
  ```

  on a Linux system you would need to escape some characters as follows:

  ```
  echo "Output('Hello world');" > hello.ecl
  ```

- Compile your program using the ECL Compiler by issuing the following command:

  ```
  eclcc hello.ecl
  ```

- An executable file is created which you can run by typing the following:

  on Linux systems:

  ```
     ./a.out
  ```

  on Windows systems:

  ```
     a.out
  ```

  This will generate the output "Hello world" (excluding quotes), to the std output, your terminal window in this example. You can redirect or pipe the output to a file or program if you choose. This simple example will verify the compiler is working properly.

Compile with Options

Once verified that the ECL Compiler is working correctly, you can try using some of the options. One such variation might be to specify the -o option which allows us to input more meaningful output filename of Hello.

```
eclcc -oHello hello.ecl
```

This produces a file called "Hello", which can now be run from the command line.

on Linux systems:

```
   ./Hello
```

on Windows systems:

```
   Hello
```

This will result in the output of the following.

```
Hello world
```

There are additional options that can be used when running the executable. Using our Hello program, as an example, we can execute it with an option to generate different output. One such option is the -xml option which generates the output in an XML format.

on Linux systems:

```
   ./Hello -xml
```

on Windows systems:

```
   Hello -xml
```

This would result in the output of the following:

```
<Dataset name="Result 1"><Row><Result_1>Hello world</Result_1></Row></Dataset>
```

The following example provides a defined value passed to the compiler:

```
//file named hello2.ecl
IMPORT ^ as repo;
OUTPUT(repo.optionXX);
```

```
eclcc -Doptionxx='HELLO' hello2.ecl
```

This would result in the output of the following:

```
<Dataset name="Result 1"><Row><Result_1>HELLO</Result_1></Row></Dataset>
```

Command Line DFU

Command Line Interface

dfuplus[--version] action=*operation* [@*filename*|*options*]

--version	displays version info
operation	One of the following actions: spray, despray, copy, remove, rename, list, add, addsuper, removesuper, listsuper, savexml, status, abort, resubmit, monitor
@filename	Optional. The name of a file containing necessary *options*. If omitted and no command line *options* are specified, the appropriate *options* must be in the dfuplus.ini file in the same directory as the executable.
options	Optional. A space-delimited list of optional items (listed below) appropriate to the *operation* being executed. If omitted and no *@filename* is specified, the appropriate *options* must be in the dfuplus.ini file in the same directory as the executable.

The **dfuplus** executable accepts command line parameters to send to the Distributed File Utility (DFU) engine via the ESP server. These *options* can be specified on the command line, in the *@filename*, in the dfuplus.ini file in the same directory as the executable, or any combination.

Evaluation of options follows this order of precedence:

- command line
- @filename file
- ini file
- default value

The dfuplus utility does not upload files to a landing zone. You must first upload any file(s) to your landing zone using either ECL Watch or a tool that supports a secure copy protocol, such as SCP or SFTP.

General Options:

The following *options* are common to every *operation*:

server	The URL (http:// or https://) and/or IP address of the ESP server. The port may also be included.
username	A userid with authorized access to the *server*.
password	The password authorizing access for the *username*.
overwrite	Optional. A boolean flag (0 | 1) indicating whether to overwrite any existing file of the same name. If omitted, the default is 0.
replicate	Optional. A boolean flag (1 | 0) indicating whether to replicate the file. If omitted, the default is 1. **This option is only available on systems where replication has been enabled.**

autorecover	Optional. The number of times to attempt recovery of a failed *operation*. If omitted, the default is 0.
nowait	Optional. A boolean flag (0 | 1) indicating whether to return immediately without waiting for completion of the *operation*. If omitted, the default is 0.
connect	Optional. The number of simultaneous connections to limit the *operation* to. If omitted, the default is 25.
throttle	Optional. The transfer speed (in Mbits/second) to restrict the *operation* to. If omitted, the default is the best system speed in Linux and multiple-destination Windows, or the NIC speed of a single-destination Windows box.
norecover	Optional. A boolean flag (0 | 1) indicating whether to create or recover the *operation* from recovery information. If omitted, the default is 0.
nosplit	Optional. A boolean flag (0 | 1) indicating whether to split file parts to multiple target parts. If omitted, the default is 0.
compress	Optional. A boolean flag (0 | 1) indicating whether to compress the target file.
push	Optional. A boolean flag (0 | 1) indicating whether to override push/pull default.
encrypt=<password>	Optional. Specifies to encrypt the target filename using the supplied password.
decrypt=<password>	Optional. Specifies to decrypt the source filename using the supplied password.
jobname=<jobname>	Specify a jobname for the DFU operation's workunit.
transferbuffersize=nnn	Optional. Overrides the DFU Server's buffer size value (default is 64k)

dfuplus.ini

Any *options* can be specified in a file called dfuplus.ini in the same directory as the executable. If your operating system is case-sensitive, make sure the filename is in lowercase. Options that rarely change can be put in the dfuplus.ini file. For example:

```
server=http://10.150.50.12:8010
username=rlor
password=password
overwrite=1
replicate=1
```

In all the examples below, we'll assume dfuplus.ini has the above content.

We do not recommend storing your password in the ini file (which is clear text). The password is included in the ini file for these examples to simplify the example code.

Spray Operations:

The **spray** *operation* copies a file from the landing zone, distributing it across all the nodes of the destination HPCC.

These *options* are used by the **spray** *operation*:

srcip	Optional. The IP address of the source machine. If omitted, the information must be supplied by the *srcxml* parameter.
srcfile	Optional. The path to the source file. This may contain wildcard characters (* and ?) to include multiple source files in the spray to a single *dstname*. If omitted, the information must be supplied by the *srcxml* parameter.
srcxml	The name of the XML file containing the information required for the *srcip* and *srcfile* parameters. This file may have been obtained by previous use of the savexml *operation*. This option provides the feature of combining multiple source files into a single resulting logical file in the HPCC.
dstname	The logical name of the destination file.
dstcluster	The name of the destination cluster.
prefix	Optional. Both of the following (separated by a comma):
filename{:*length***}**	Prepends the filename (optionally limited to *length* characters) to the data.
filesize{:[B|L][1-8]}	Prepends the size of the file to the data. Optionally, you can specify the format of that integer (**B** specifies big endian, **L** specifies little endian) and the size of integer to contain it (**1** to **8** bytes). If format and size are omitted, the default is L4.
format	Optional. One of the following values: **fixed csv delimited xml recfmv recfmb** If omitted, the default is fixed.
fixed *format* options:	
recordsize	The fixed size of each record, in bytes.
csv/delimited *format* options:	
encoding	Optional. One of the following: ascii, utf8, utf8n, utf16, utf16le, utf16be, utf32, utf32le, utf32be ; If omitted, the default is ascii.
maxrecordsize	Optional. The maximum size of each record, in bytes. If omitted, the default is 8192.
separator	Optional. The field delimiter. If omitted, the default is a comma (\,).

terminator	Optional. The record delimiter. If omitted, the default is line feed or carriage return line feed (\r,\r\n).
quote	Optional. The string quote character. If omitted, the default is single quote (’).
xml *format* options:	
rowtag	The XML tag identifying each record.
encoding	Optional. One of the following: utf8 utf8n utf16 utf16le utf16be utf32 utf32le utf32beIf omitted, the default is utf8.
maxrecordsize	Optional. The maximum size of each record, in bytes. If omitted, the default is 8192.

Examples:

```
//fixed spray example:
dfuplus action=spray srcip=10.150.50.14
          srcfile=c:\import\timezones.txt dstname=RTTEMP::timezones.txt
          dstcluster=thor format=fixed recordsize=155

//fixed spray example using a srcxml file:
dfuplus action=spray srcxml=c:\import\flattimezones.xml
          dstname=RTTEMP::timezones.txt dstcluster=thor recordsize=155

//csv spray example:
dfuplus action=spray srcip=10.150.50.14
          srcfile=c:\import\timezones.csv dstname=RTTEMP::timezones.csv
          dstcluster=thor format=csv

//the spray.xml file contains:
<File directory="c:\import\"
   group="thor"
   modified="2004-04-27T14:58:38"
   name="zip"
   numparts="2"
   partmask="zip._$P$_of_$N$">
<Attr job="zip1"
   owner="rtaylor"
   recordSize="5"
   replicated="1"
   workunit="D20040427-111857"/>
<Part modified="2004-04-27T14:58:40"
   node="10.150.51.29"
   num="1"
   size="165"/>
<Part modified="2004-04-27T14:58:40"
   node="10.150.51.29"
   num="2"
   size="165"/>
</File>

//fixed spray example using the above spray.xml file to
          combine
// multiple source files into a single logical file
// in this case, zip._1_of_3, zip._2_of_3, and zip._3_of_3
          into zip1:
dfuplus action=spray srcxml=spray.xml
          dstcluster=thordstname=RTTEMP::myzip1 recordsize=5

//xml spray example:
dfuplus action=spray srcip=10.150.50.14
          srcfile=c:\import\timezones.xml dstname=RTTEMP::timezones.xml
          dstcluster=thor format=xml rowtag=area

//Multiple spray all .JPG and .BMP files under
```

```
// c:\import on 10.150.51.26 to single logical file
          LE::imagedb:
dfuplus action=spray srcip=10.150.51.26
          srcfile=c:\import\*.jpg,c:\import\*.bmp

dstcluster=le_thor dstname=LE::imagedb overwrite=1
          prefix=FILENAME,FILESIZE nosplit=1
//this would result in a RECORD structure like this:
imageRecord := RECORD
STRING filename;
DATA image; //first 4 bytes contain the length of the image data
END;
```

Despray Operations:

The **despray** *operation* combines file parts from all the nodes of the cluster into a single file on the landing zone.

These *options* are used by the **despray** operation:

srcname	The logical name of the source file. This may contain wildcard characters (* and ?) to include multiple source files in the despray to a single *dstfile*.
dstip	Optional. The IP address of the destination machine. If omitted, the information must be supplied by the *dstxml* parameter.
dstfile	Optional. The path to the destination file. This may contain wildcard characters (* and ?) to despray a single *srcname*to multiple *dstfiles*. If omitted, the information must be supplied by the *dstxml* parameter.
dstxml	The name of the XML file containing the information required for the *dstip* and *dstfile* parameters. This file may have been obtained by previous use of the savexml *operation*. This option provides the feature of splitting a single resulting logical file in the cluster into multiple destination files.
splitprefix	Optional. Both of the following (separated by a comma):
filename{:*length***}**	Uses the prepended filename (see the *prefix* option to the spray *operation*) to split out the data into separate files.
filesize{:[B|L][1-8]}	Uses the prepended size of the file (see the *prefix* option to the spray *operation*) to split out the data into separate files.

Examples:

```
dfuplus action=despray dstip=10.150.50.14
   dstfile=c:\import\despray\timezones.txt srcname=RTTEMP::timezones.txt
//the spray.xml file contains:
<File directory="c:\import\"
   group="thor"
   modified="2004-04-27T14:58:38"
   name="zip"
   numparts="2"
   partmask="zip._$P$_of_$N$">
<Attr job="zip1"
   owner="rtaylor"
   recordSize="5"
   replicated="1"
   workunit="D20040427-111857"/>
<Part modified="2004-04-27T14:58:40"
   node="10.150.51.29"
   num="1"
   size="165"/>
<Part modified="2004-04-27T14:58:40"
```

```
   node="10.150.51.29"
   num="2"
size="165"/>
</File>
//despray example using the above spray.xml file to split a single
// logical file into multiple destination files
// in this case, zip._1_of_3, zip._2_of_3, and zip._3_of_3 from zip1:
dfuplus action=despray dstxml=spray.xml dstcluster=thor
          srcname=RTTEMP::myzip1


//from a RECORD structure that looks like this:
imageRecord := RECORD
STRING filename;
DATA image; //first 4 bytes contain the length of the image data
          END;

//you can despray into its component files like this:
dfuplus action=dspray srcname=le::imagedb
          dstip=10.150.51.26 dstfile=c:\export\
          splitprefix=FILENAME,FILESIZE
```

Copy Operations:

The **copy** *operation* copies a logical file (all file parts from all the nodes of the cluster), typically from one cluster to another. It appropriately handles re-distributing the file parts if the source and destination clusters do not have the same number of nodes.

The copy operation can also be used to copy files from other HPCC environments (using the *srcdali* option). This is also known as a remote copy. For a remote copy of a file that contains a variable length field, you must include the **nosplit** option.

These *options* are used by the **copy** *operation*:

srcname	The logical name of the source file.
dstname	The logical name of the destination file.
dstcluster	The name of the destination cluster.
srcdali	Optional. The IP address of the source Dali server, if different from the destination Dali (associated with the ESP Server specified in the *server* option).
srcusername	Optional. The username to use to access the *srcdali*. If omitted, the General Options *username* is used.
srcpassword	Optional. The password to use to access the *srcdali*. If omitted, the General Options *password* is used.
preservecompression	Optional. A boolean flag (0 | 1) indicating whether to preserve the compression of the source file. If omitted, the default is 1.

Example:

```
dfuplus action=copy srcname=RTTEMP::timezones.txt
          dstname=srcname=RTTEMP::COPY::timezones.txt dstcluster=thor
```

Remove Operations:

The **remove** operation deletes a logical file from the system data store, optionally leavingthe physical files in place.

These *options* are used by the **remove** *operation*:

name	The logical name of the file to remove.

Example:

```
dfuplus action=remove name=RTTEMP::timezones.txt
```

Rename Operations:

The **rename** operation renames a logical file in the system data store.

These *options* are used by the **rename** *operation*:

srcname	The logical name of the source file.
dstname	The logical name of the destination file.

Example:

```
dfuplus action=rename srcname=RTTEMP::timezones.txt dstname=RTTEMP::NewTimezones.txt
```

List Operations:

The **list** operation produces a list of logical files in the system data store.

These *options* are used by the **list** *operation*:

name	The mask defining the logical file names to list.

Example:

```
dfuplus action=list name=*
```

Add Operations:

The **add** operation adds a new logical file to the system data store.

These *options* are used by the **add** *operation*:

srcxml	The logical name of the source XML file map (typically from a previous savexml operation).
dstname	The logical name of the destination file.

These *options* are used by the **add** *operation* to add files from a remote Dali:

dstname	The logical name of the destination file.
srcname	The logical name of the source file.
srcdali	The IP address of the source Dali server.
srcusername	Optional. The username to use to access the *srcdali*. If omitted, the General Options *username* is used.
srcpassword	Optional. The password to use to access the *srcdali*. If omitted, the General Options *password* is used.

Example:

```
dfuplus action=add srcxml=flattimezones.xml dstname=flattimezones.txt
```

Addsuper Operations:

The **addsuper** operation adds subfiles to an existing superfile (see the *SuperFile Management* section of the *Service Library Reference*).

These *options* are used by the **addsuper** *operation*:

superfile	The logical name of the superfile.
subfiles	A comma-delimited list of the logical names of files to add to the superfile. There must be no spaces between the names.
before	Optional. The logical name of the subfile to follow the added *subfiles*. If omitted, the *subfiles* are added to the end.

Example:

```
dfuplus action=addsuper superfile=mysuper subfiles=file1,file2
```

Removesuper Operations:

The **removesuper** operation removes subfiles to an existing superfile (see the *SuperFile Management* section of the *Service Library Reference*).

These *options* are used by the **removesuper** *operation*:

superfile	The logical name of the superfile.
subfiles	Optional. A comma-delimited list of the logical names of files to remove from the superfile. There must be no spaces between the names. If omitted, all files are removed from the superfile.
delete	Optional. A boolean flag (1 | 0) indicating whether to physically delete the *subfiles* in addition to removing them from the superfile. If omitted, the default is 1—physically delete.

Example:

```
dfuplus action=removesuper superfile=mysuper subfiles=file1,file2
```

Listsuper Operations:

The **listsuper** operation lists the subfiles in an existing superfile (see the *SuperFile Management* section of the *Service Library Reference*).

These *options* are used by the **listsuper** *operation*:

superfile	The logical name of the superfile.

Example:

```
dfuplus action=listsuper superfile=mysuper
```

Status Operations:

The **status** operation returns the current operational status of a workunit.

These *options* are used by the **status** *operation*:

wuid	The workunit identifier of the workunit.

Example:

```
dfuplus action=status wuid=W20050309-093020
```

Abort Operations:

The **abort** operation aborts execution of a workunit.

These *options* are used by the **abort** *operation*:

wuid	The workunit identifier of the workunit.

Example:

```
dfuplus action=abort wuid=W20050309-093020
```

Resubmit Operations:

The **resubmit** operation re-submits a workunit.

These *options* are used by the **resubmit** *operation*:

wuid	The workunit identifier of the workunit.

Example:

```
dfuplus action=resubmit wuid=W20050309-093020
```

Savexml Operations:

The **savexml** operation saves the logical file map to an XML file.

These *options* are used by the **savexml** *operation*:

srcname The logical name of the source file.

srcname	The logical name of the source file.
dstxml	Optional. The logical name of the destination XML file. If omitted, the XML result is sent to stdout.

Example:

```
dfuplus action=savexml srcname=RTTEMP::timezones.txt
          dstxml=flattimezones.xml
    // this results in the following XML file:
    <File directory="c:\thordata\rttemp"
          group="thor"
          modified="2004-06-18T14:17:16"
          name="timezones.txt"
          numparts="3"
```

```
      partmask="timezones.txt._$P$_of_$N$">
<Attr job="timezones.txt"
      owner="rtaylor"
      recordSize="155"
      replicated="1"
      size="51305"
      workunit="D20040618-101716"/>
<OrigName>rttemp::timezones.txt</OrigName>
<Part modified="2004-06-18T14:17:18"
      node="10.150.50.15"
      num="1"
      size="17050"/>
<Part modified="2004-06-18T14:17:17"
      node="10.150.50.18"
      num="2"
      size="17050"/>
<Part modified="2004-06-18T14:17:17"
      node="10.150.50.16"
      num="3"
      size="17205"/>
</File>
```

Monitor Operations:

The **monitor** operation initiates a DFU workunit to monitor the appearance of a physical or logical file and trigger an event when that file appears.

These *options* are used by the **monitor** *operation*:

event	The name of the user-defined event to trigger. This is used as the first parameter of the ECL EVENT function.
lfn	Optional. The name of the logical file in the DFU to look for. Using this option precludes using the *ip*, *file*, and *sub* options.
ip	Optional. The IP address or name of the server on which the physical file will reside. This may be omitted if the *file* option contains a full URL.
file	Optional. The fully qualified path of the physical file to look for. This may contain wildcard characters (* and ?).
sub	Optional. Specifies searching subdirectories for the physical file if the *file* option contains wildcard characters (* and ?).
shotlimit	Optional. The number of arrival events to generate before marking the DFU workunit as complete. A value of negative one (-1) indicates continuing until the workunit is manually aborted. If omitted, the default value is one (1).

Note the following caveats and restrictions:

1) If a matching file already exists when the DFU Monitoring job is started, that file will not generate an event. It will only generate an event once the file has been deleted and recreated.

2) If a file is created and then deleted (or deleted then re-created) between polling intervals, it will not be seen by the monitor and will not trigger an event.

3) Events are only generated on the polling interval.

4) Note that the *event* is generated if the physical file has been created since the last polling interval. Therefore, the *event* may occur before the file is closed and the data all written. To ensure the file is not subsequently read before it is complete you should use a technique that will preclude this possibillity, such as using a separate 'flag' file instead of the file, itself or renaming the file once it has been created and completely written.

5) The EVENT function's subtype parameter (its 2nd parameter) when monitoring physical files is the full URL of the file, with an absolute IP rather than DNS/netbios name of the file. This parameter cannot be retrieved but can only be used for matching a particular value in this.

Example:

```
dfuplus action=monitor event=MyEvent ip=edata10 file=/dz/arr.txt
dfuplus action=monitor event=MyEvent ip=10.150.10.75
           file=c:\dz\* shotlimit=-1 sub=1
dfuplus action=monitor event=MyEvent file=//10.15.13.21/dz/*.txt
dfuplus action=monitor event=MyEvent lfn=RTTEMP::OUT::MyFile
```

ESDL Command Line Interface

The ESDL Command Syntax

esdl [--version] <command> [<options>]

--version	displays version info.
help <commmand>	displays help for the specified command.
xml	Generate XML from ESDL definition.
ecl	Generate ECL from ESDL definition.
xsd	Generate XSD from ESDL definition.
wsdl	Generate WSDL from ESDL definition.
publish	Publish ESDL Definition for ESP use.
list-definitions	List all ESDL definitions.
delete	Delete ESDL Definition.
bind-service	Configure ESDL based service on target ESP (with existing ESP Binding).
list-bindings	List all ESDL bindings.
unbind-service	Remove ESDL based service binding on target ESP.
bind-method	Configure method associated with existing ESDL binding.
get-binding	Get ESDL binding.

esdl xml

esdl xml [options] filename.ecm [<outdir>]

filename.ecm	The file containing the ESDL definitions
-r|--recursive	process all includes
-v|--verbose	display verbose information
-?/-h/--help	show usage page
Output	(srcdir|<outdir>)/filename.xml

This generates XML from the ESDL definition. This XML is an intermediate entity used by the ESDL Engine to create the runtime service definitions. This command is rarely used by itself.

Examples:

```
esdl xml MathService.ecm .
```

esdl ecl

esdl ecl [options] filename.ecm [<outdir>]

filename.ecm	The file containing the ESDL definitions
-r|--recursive	process all includes
-v|--verbose	display verbose information
-?/-h/--help	show usage page
Output	(srcdir|<outdir>)/filename.ecl

This generates ECL structures from ESDL definition. These structures create the interface (entry and exit points) to the Roxie query.

Examples:

```
esdl ecl MathService.ecm .
```

esdl xsd

esdl xsd [options] filename.ecm [<outdir>]

filename.ecm	The file containing the ESDL definitions
-r|--recursive	process all includes
-v|--verbose	display verbose information
-?/-h/--help	show usage page
Output	(srcdir|<outdir>)/filename.ecl

This generates XSD from the ESDL definition.

Examples:

```
esdl xsd MathService.ecm .
```

esdl wsdl

esdl wsdl [options] filename.ecm [<outdir>]

filename.ecm	The file containing the ESDL definitions
-r|--recursive	process all includes
-v|--verbose	display verbose information
-?/-h/--help	show usage page
Output	(srcdir|<outdir>)/filename.ecl

This generates WSDL from ESDL definition.

Examples:

```
esdl wsdl MathService.ecm .
```

esdl publish

esdl publish <servicename> <filename.ecm>)[options]

servicename	The name of the service to publish
filename.ecm	The ESDL (*.ecm) file containing the service definitions.
--overwrite	Overwrite the latest version of this ESDL Definition
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Publishes an ESDL service definition to the system datastore.

Examples:

```
esdl publish mathservice mathservice.ecm -s nnn.nnn.nnn.nnn --port 8010
```

esdl list-definitions

esdl list-definitions [options]

-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

This command lists published definitions

Example:

```
esdl list-definitions -s nnn.nnn.nnn.nnn --port 8010
```

esdl delete

esdl delete <ESDLServiceDefinitionName> <ESDLServiceDefinitionVersion> [options]

ESDLServiceDefinitionName	The name of the ESDL service definition to delete
ESDLServiceDefinitionVersion	The version of the ESDL service definition to delete
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Use this command to delete an ESDL Service definition. If the Service definition is bound, you must first unbind it.

Example:

```
esdl delete mathservice 2 -s nnn.nnn.nnn.nnn --port 8010
```

esdl bind-service

esdl bind-service

esdl bind-service <TargetESPProcessName> <TargetESPBindingPort | TargetESPServic eName> <ESDLDefinitionId> <ESDLServiceName> [command options]

TargetESPProcessName	The target ESP Process name
TargetESPBindingPort | TargetESPServiceName	Either target ESP binding port or the target ESP service name
ESDLDefinitionId	The Name and version of the ESDL definition to bind to this service (must already be defined in Dali)
ESDLServiceName	The Name of the ESDL Service (as defined in the ESDL Definition)
--config <file | XML>	Configuration XML (either inline or as a file reference)
--overwrite	Overwrite the latest version of this ESDL Definition
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Use this command to bind a Dynamic ESDL-based ESP service to an ESDL definition.

To bind an ESDL Service, provide the target ESP process name (ESP Process which will host the ESP Service as defined in the ESDL Definition.)

You must also provide the Port on which this service is configured to run (ESP Binding), and the name of the service you are binding.

Optionally provide configuration information either directly inline or using a configuration file XML in the following syntax:

```
<Methods>
  <Method name="myMthd1" url="http://<RoxieIP>:9876/somepath?someparam=value" user="me" password="mypw"/>
  <Method name="myMthd2" url="http://<RoxieIP>:9876/somepath?someparam=value" user="me" password="mypw"/>
</Methods>
```

Example:

```
esdl bind-service myesp 8003 MathSvc.1 MathSvc --config MathSvcCfg.xml -s nnn.nnn.nnn.nnn -p 8010
```

esdl list-bindings

esdl list-bindings [options]

-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Use this command to list bindings on a server.

Example:

```
esdl list-bindings -s nnn.nnn.nnn.nnn -p 8010
```

esdl unbind-service

esdl unbind-service <TargetESPProcessName> <TargetESPBindingPort | TargetESPServ iceName> [options]

TargetESPProcessName	The target ESP Process name
TargetESPBindingPort | TargetESPServiceName	Either target ESP binding port or the target ESP service name
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Use this command to unpublish ESDL Service based bindings.

To unbind an ESDL Service, provide the target ESP process name (ESP Process which will host the ESP Service as defined in the ESDL Definition.) You must also provide the Port on which this service is configured to run (the ESP Binding), and the name of the service you are unbinding.

Example:

```
esdl unbind-service myesp 8003
```

esdl bind-method

esdl bind-method <TargetESPProcessName> <TargetESPBindingName> <TargetServiceName> <TargetServiceDefVersion> <TargetMethodName> [options]

TargetESPProcessName	The target ESP Process name
TargetESPBindingName	Either target ESP binding name
TargetServiceName	The name of the Service to bind (must already be defined in dali.)
TargetServiceDefVersion	The version of the target service ESDL definition (must exist in dali)
TargetMethodName	The name of the target method (must exist in the service ESDL definition)
--config <file | XML>	Configuration XML (either inline or as a file reference)
--overwrite	Overwrite the latest version of this ESDL Definition
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Use this command to publish ESDL Service based bindings.

To bind an ESDL Service, provide the target ESP process name (ESP Process which will host the ESP Service as defined in the ESDL Definition.)

You must also provide the Port on which this service is configured to run (ESP Binding), and the name of the service you are binding.

Optionally provide configuration information either directly inline or using a configuration file XML in the following syntax:

```
<Methods>
  <Method name="myMthd1" url="http://<RoxieIP>:9876/somepath?someparam=value" user="me" password="mypw"/>
  <Method name="myMthd2" url="http://<RoxieIP>:9876/somepath?someparam=value" user="me" password="mypw"/>
</Methods>
```

Example:

```
esdl bind-service myesp 8003 MathSvc.1 MathSvc --config MathSvcCfg.xml -s nnn.nnn.nnn.nnn -p 8010
```

esdl get-binding

esdl get-binding <ESDLBindingId> [options]

ESDLBindingId	The target ESDL binding id <espprocessname>.<espbindingname>
-s, --server	The IP Address or hostname of ESP server running ECL Watch services
--port	The ECL Watch services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary)
--version <ver>	ESDL service version
--help	display usage information for the given command
-v, --verbose	Output additional tracing information

Use this command to get DESDL Service based bindings.

To specify the target DESDL based service configuration, provide the target ESP process (esp process name or machine IP Address) which hosts the service.

You must also provide the Port on which this service is configured to run and the name of the service.

Example:

```
esdl get-binding  myesp.dESDL_Service -s nnn.nnn.nnn.nnn -p 8010
```