

# HPCC Systems® Client Tools

Boca Raton Documentation Team



## HPCC Systems® Client Tools

Boca Raton Documentation Team

Copyright © 2025 HPCC Systems®. All rights reserved

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.

LexisNexis and the Knowledge Burst logo are registered trademarks of Reed Elsevier Properties Inc., used under license.

HPCC Systems® is a registered trademark of LexisNexis Risk Data Management Inc.

Other products, logos, and services may be trademarks or registered trademarks of their respective companies.

All names and example data used in this manual are fictitious. Any similarity to actual persons, living or dead, is purely coincidental.

2025 Version 9.10.38-1

Overview .....	4
Documentation Conventions .....	5
ECL Command Line Interface .....	8
The ECL Command Syntax .....	8
ECL Compiler .....	73
Using the ECL Compiler as a Stand Alone option .....	74
Compiled Options: .....	78
Examples .....	79
Command Line DFU .....	81
Command Line Interface .....	81
ESDL Command Line Interface .....	94
The ESDL Command Syntax .....	94

# Overview

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

<b>ECL</b>	Command line ECL tool
<b>ECL Compiler</b>	Command line ECL Compiler
<b>DFUPlus</b>	Command line Distributed File Utility management tool, facilitate automation of data file spray, despray, and other common file handling tasks.
<b>ESDL</b>	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 all caps, such as: DOUBLE-CLICK, or press the ENTER keyword.
- 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 DFUPlus and the ECL Command line tools.

1. From the HPCC Systems® download page, <https://hpccsystems.com/download>

Download the appropriate Client Tools for your Operating System. (available for RPM-Based systems, Debian-Based systems, Mac OSX, or Windows)

2. Install the client tools software to your machine.

### Windows:

Run the executable file, for example: hpccsystems-clienttools\_community-7.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>
```

After installing the package, run the following to "fix" any missing dependencies:

```
sudo apt-get install -f
```

### Mac OSX:

Open the Apple disk image file (.dmg) and then run the installation package (.pkg). Follow the prompts to complete the installation.

## Multiple Version Installations

It is possible to install multiple versions of the client tools if you need to work with multiple versions of the platform.

To install the client tools, obtain the appropriate installation package for your operating system and the version to match your HPCC Systems server:

1. Download the appropriate Client Tools for your Operating System and version.

Client tools can be found at the HPCC Systems® download page:

<https://hpccsystems.com/download>

**NOTE:** There is a link at the bottom of the list "[view older downloads](#)" if you are looking for previous versions.

2. Install the Client Tools on to your system. Take note of the following considerations:

Client tool packages starting with 4.2 have built in logic to allow for multiple installations. Prior versions of the client tools package would just overwrite the existing components. The default behavior is that the client

tools will use the last one installed, except if you are working directly on the platform. If you are working directly on the platform then it would use the client tools package that gets installed with the platform.

If you install a version other than the delivered client tools you will have a folder in /opt/HPCCSystems that corresponds to the set of client tools. So you could have a client tools 7.0.x, 7.2.x, 7.4.x, etc.

For older versions, download the package(s), and install. Install the one you want to use last. Copy to a different folder or Rename the client tools found in /opt/HPCCSystems after installing the older version and before installing the newer version. This is to prevent the newer client tools from overwriting the older one.

To use the Client tools for the various version number(s) explicitly call the client tool you wish to use, or set up an alias to call the client tool using the proper path or name for the version you intend to use. This would depend on how you chose to save off the older client tools you installed.

**For example**, if you wanted to run DFUplus:

```
dfuplus action=list server=http://127.0.0.1:8010
```

To run DFUplus for an older or another version of client tools, for instance 7.0.x:

```
/opt/HPCCSystems/7.0.x/clienttools/bin/dfuplus action=list server=http://127.0.0.1:8010
```

## Windows

Client tools for Windows installs in a directory such as: C:\Program Files (x86)\HPCCSystems\7.2.0\client-tools\bin where the number (7.2.0 for example) corresponds to the version of the client tools.

If you want access to one version of the command line client tools from any folder, you can add the \bin folder to your Path in Windows (for example, **C:\Program Files (x86)\HPCCSystems\7.2.0\clienttools\bin** )

The Windows installer will prompt you to delete the previous version during installation. If you want to keep both, decline the offer to uninstall, and choose a different installation directory at the next prompt.

# ECL Command Line Interface

## The ECL Command Syntax

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

<i>--version</i>	displays version info.
<b>Arguments</b>	
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
results	returns the full results of a given WUID in XML format.
activate	Activate a published query
deactivate	Deactivate the given query alias name
queries	List or manipulate queries and querysets
packagemap	execute packagemap commands (for Roxie)
bundle	manage ECL bundles
roxie	execute commands for Roxie
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.
zapgen	generate and download the zap file for the given WUID.
sign	adds a digital signature to an ecl or text file.
listkeyuid	returns a list of all the key user IDs that can be used by the sign command.
url-secret-name	generate a secret name from a URL for automatic URL mapping



## 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:

```
;The values below are examples, you should change them to match your platform deployment
eclWatchIP=10.150.50.12
eclWatchPort=28010
eclUserName=emilykate
eclPassword=
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 or environment variable (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

If a username is supplied by any method, and a password is not, you are prompted for the password.

## 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
ECLCC_PATH
```



We do not recommend storing your password in an Environment Variable.

## Git Support

HPCC Systems integrates native support for leveraging Git.

The `--main` option has been extended to support compiling a query directly from a Git repository. When invoked, it retrieves the ECL code specified from the Git repository, compiles the code, and runs the query. The checkout is done on the remote ECLCCServer rather than on the client machine.

## Syntax

The syntax to reference a repository location:

```
<protocol>://<urn>/<user>/<repository>#version
```

The `#version` component can be either the name of a *branch*, *tag*, or the SHA (Secure Hash Algorithm) of a *commit*.

For example:

```
ecl run thor --main demo.main@gituser/gch-ecldemo-d#version1 --server=...
```

This example runs the *demo.main* ECL code in the *version1* branch of the *gituser/gch-ecldemo-d* repository.

This feature is supported by both the `ecl` and `eclcc` command line tools.

## Additional Examples

The following examples all compile the attribute *demo.main* from *version3* branch of the GitHub repository *gituser/gch-ecldemo-d*:

```
eclcc --main demo.main@gituser/gch-ecldemo-d#version3
eclcc --main demo.main@gituser/gch-ecldemo-d --mainrepoversion=version3
eclcc --main demo.main@gituser/gch-ecldemo-d --defaultrepoversion=version3
eclcc --main demo.main#version3 --defaultrepo=gituser/gch-ecldemo-d
eclcc --main demo.main --defaultrepo=gituser/gch-ecldemo-d --defaultrepoversion=version3
```

If you are submitting a source file *query.ecl*, and you want it to replace the definition of *demo.main* with the same repo, the following will work:

```
eclcc query.ecl --main demo.main@gituser/gch-ecldemo-d#version3
eclcc query.ecl --main demo.main@gituser/gch-ecldemo-d --mainrepoversion=version3
eclcc query.ecl --main "demo.main#version3" --mainrepo=gituser/gch-ecldemo-d
eclcc query.ecl --main demo.main --defaultrepo=gituser/gch-ecldemo-d --defaultrepoversion=version3
```

## Multiple Repository Support

Queries can be compiled from multiple Git repositories. Each Git repository is treated as a separate independent package. Dependencies between the repositories are specified in a package file which is checked into the repository and versioned along with the ECL code. The package file indicates what the dependencies are and which versions should be used.

```
package.json:
{
  "name": "demoRepoC",
  "version": "1.0.0",
  "dependencies": {
    "demoRepoD": "gituser/gch-ecldemo-d#version1"
  }
}
```

In the above example, the package file gives a *name* to the package and defines the *dependencies*.

The dependencies property is a list of key-value pairs. The key provides the name of the ECL module that is used to access the external repository. The value is a repository reference using the same syntax described in the previous section.

## External Repositories in ECL

To use definitions from an external repository you would add an import definition to your ECL code.

For example consider the following ECL:

```
IMPORT layout;  
IMPORT demoRepoD AS demoD;  
EXPORT personAsText(layout.person input) :=  
  input.name + ': ' + demoD.format.maskPassword(input.password);
```

The name demoRepoD in the second import matches the key value in the package.json file.

This code uses the attribute *format.maskPassword* from the repository gituser/gch-ecl-demo-d.

Each package is processed independently of any others. The only connection is through explicit imports of the external packages. This means that packages can have modules or attributes with the same name and they will not clash.

## The NPM

You can use the NPM (node package manager) to ensure that labels or branches are tied down to a specific SHA.

For example :

```
npm install --package-lock-only
```

This command creates a *package-lock.json* file in the same location as *package.json*. The npm program then resolves the references to branches and resolves them to the corresponding SHAs. Using the *--package-lock-only* option indicates that the npm should not clone the associated versions of the code to *node\_modules* directories.

The generated *package-lock.json* file will contain something similar to the following:

```
{  
  "packages": {  
    "node_modules/demoRepoD": {  
      "resolved": "https://github.com/gituser/gch-ecl-demo-d.git#644clf4dd80ca1e8f05974983455a244e5",  
    }  
  }  
}
```

If a *package-lock.json* file is present it will take precedence over the *package.json* file. The ECL is compiled in the same way, with ECLCC Server automatically downloading the dependencies.

The advantage of using *package-lock.json* over *package.json* is that it allows you to use npm's semantic versioning syntax (*#semver*). It also allows you to use a branch in your *package.json* file as a logical dependency, but resolved to an actual dependency or specific SHA. That way if the branch is updated the query will not change.

You can also use npm without the *--package-lock-only* option. This will check out the appropriate version of the code into the *node\_modules* subdirectory of the current project. The ECLCC Server supports the

node\_module structure as a way of providing the source for external packages. This is an alternative way to compile the code using eclcc completely independently from the source control system.

## ecl deploy

**ecl deploy <target> <file> [--job-name=<value>]**

**ecl deploy <target> <archive> [--job-name=<value>]**

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

**ecl deploy <target> - [--job-name=<val>]**

Examples:

```
ecl deploy roxie findperson.ecl --job-name=FindPersonService
ecl deploy roxie ArchiveQuery.xml --job-name=FindPersonService
ecl deploy roxie libW20150914-125557.so --job-name=FindPersonService
ecl deploy roxie - --job-name=FindPersonService
```

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

ecl deploy	Creates a workunit on the HPCC Systems platform from the given ECL text, file, archive, shared object, or dll. The workunit is created in the <i>compiled</i> state.
<b>Arguments</b>	
target	The target cluster to which to deploy
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
-	Specifies object should be read from stdin
<b>Options</b>	
--job-name	The published query name
--protect	Protect the workunit from deletion
<b>ECL Options</b>	
--main=<definition>	This option allows you to compile queries from existing source trees or repositories. It can be used to specify default repositories and versions to use. See the "Git Support" section for specific usage.
--snapshot, -sn=<label>	Snapshot label to use from a legacy ECL repository
--ecl-only	Send ECL query to HPCC Systems cluster as text rather than gathering dependencies and building an archive
--limit=<limit>	Sets the result limit for the query
-f<option>[=value]	Set an ECL option (equivalent to #option)
-f-<option>[=value]	Set an eclcc command line option with the single dash ('-')
-f--<option>[=value]	Set an eclcc command line option with the double dash ('--')
-Dname=value	Override the definition of a global attribute 'name'
<b>eclcc Options</b>	
-I <path>	Add path to locations to search for ecl imports
-L <path>	Add path to locations to search for system libraries
--manifest	Specify path to the manifest file
-g, --debug	Enable debug symbols in generated code

-checkDirty	This option reports any modified attributes using git status
--fetchrepos	This option automatically downloads any missing repositories associated with dependencies
-R<repo>[#version]=<path>	This option resolves repository references specified in directory <path>
--updaterepos	Setting this option automatically updates repositories associated with dependencies
<b>Common Options</b>	
--help	Displays usage information for the given command
-v, --verbose	Output additional tracing information
-s, --server=<ip>	The IP Address or hostname of ESP server running ECL services
--port=<port>	The ECL services port (Default is 8010)
-ssl, --ssl	Use SSL to secure the connection to the server.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username=<name>	The username for accessing ECL services
-pw, --password	The password for accessing ECL services. If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl publish

**ecl publish <target> <file> [--query-name=<val>]**

**ecl publish <target> <file> [--query-name=<val>] [--job-name=<val>]**

**ecl publish <target> <wuid> [--query-name=<val>]**

**ecl publish <target> <so | dll> [--query-name=<val>]**

**ecl publish <target> <archive> [--query-name=<val>]**

**ecl publish <target> - [--query-name=<val>]**

Examples:

```
ecl publish roxie findperson.ecl --query-name=FindPersonService -A
ecl publish roxie W20150914-125557 --query-name=FindPersonService -A
ecl publish roxie libW20150914-125557.so --query-name=FindPersonService -A
ecl publish roxie ArchiveQuery.xml --query-name=FindPersonService -A
ecl publish roxie - --query-name=FindPersonService --activate
ecl publish roxie findperson.ecl --query-name=FindPersonService --no-activate
ecl publish roxie ArchiveQuery.xml --query-name=FindPersonService --no-activate
```

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.
<b>Arguments</b>	
target	The target cluster to which to publish
wuid	The workunit id to publish (WUID is case-sensitive)
file	The ECL text file to publish
archive	The ECL archive to publish
so   dll	The workunit dynamic linked library or shared object to publish
-	Specifies object should be read from stdin
<b>Options</b>	
--query-name	The query name to use for published workunit
--job-name	The job name to use for the workunit
-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
--protect	Protects the workunit specified from deletion.
--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.
--daliip=<IP>	IP address or hostname of the remote Dali to use for remote logical file lookups.

-O, --overwrite	Completely replaces the existing DFS file information. This option is dangerous - only use as a last resort.
--update-super-files	Update local DFS super-files if remote DALI has changed
--update-clone-from	Update local clone from location if remote DALI has changed
--dont-append-cluster	Issue this option only to avoid locking issues due to adding a cluster to the file
---source-process	Process cluster from which to copy files
--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.
--priority=<val>	The priority for this query. Value can be LOW, HIGH, SLA, NONE. NONE will clear current setting.
--comment=<string>	A comment associated with this query
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>ECL Options</b>	
--main=<definition>	This option allows you to compile queries from existing source trees or repositories. It can be used to specify default repositories and versions to use. See the "Git Support" section for specific usage.
--snapshot, -sn=<label>	Snapshot label to use from a legacy ECL repository
--ecl-only	Send ECL query to HPCC Systems cluster as text rather than as a generated archive
--limit=<limit>	Sets the result limit for the query
-f<option>[=value]	Set an ECL option (equivalent to #option)
-f-<option>[=value]	Set an eclcc command line option with the single dash ('-')
-f--<option>[=value]	Set an eclcc command line option with the double dash ('--')
-Dname=value	Override the definition of a global attribute 'name'
<b>eclcc Options</b>	
-I <path>	Add path to locations to search for ecl imports



HPCC Systems® Client Tools  
ECL Command Line Interface

-L <path>	Add path to locations to search for system libraries
--manifest	Specify path to the manifest file
-g, --debug	Enable debug symbols in generated code
-checkDirty	This option reports any modified attributes using git status
--fetchrepos	This option automatically downloads any missing repositories associated with dependencies
-R<repo>[#version]=<path>	This option resolves repository references specified in directory <path>
--updaterepos	Setting this option automatically updates repositories associated with dependencies
--nostdinc	Do not include the current directory in -I
--fastsyntax	Delay expanding functions when parsing. May speed up processing for some queries
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username=<name>	The username for accessing ECL services
-pw, --password	The password for accessing ECL services. If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl unpublsh

**ecl unpublsh <target> <query\_id>**

### Example:

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

ecl unpublsh	executes the supplied ecl unpublsh command
<b>Options</b>	
<target>	The name of target queryset containing the query to remove
<query_id>	The query to remove from query set. Wildcards allowed, but must be in quotes (e.g., "MyQuery*").
--activated=yes no	Specify yes to unpublsh activated queries or no to unpublsh deactivated queries. When the <query_id> is '*'. Default to all queries.
--suspended-by-user	Unpublsh the queries suspended by user when <query_id> is '*'.
--delete-workunit	Delete the workunit for the <query_id>. Default to No.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl run

**ecl run <target> <file> [--job-name=<val>] [--input=<file|xml>] [--wait=<i>]**

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

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

**ecl run <target> <so | dll> [--job-name=<val>][--input=<file|xml>][--wait=<i>]**

**ecl run <target> <archive> --job-name=<val> [--input=<file|xml>][--wait=<i>]**

**ecl run <target> - --job-name=<val> [--input=<file|xml>][--wait=<i>]**

### Examples:

```
ecl run thor findperson.ecl --job-name=findperson --input=data.xml --wait=1000
ecl run thor W20150914-125557 --input=data.xml --wait=1000
ecl run thor findperson --input=data.xml --wait=1000
ecl run thor libW20150914-125557.so --input=data.xml --wait=1000
ecl run thor - --input=data.xml --poll --wait=1000
ecl run thor findperson.ecl --input="<request><LName>JONES</LName></request>"
ecl run thor findperson.ecl -I C:\MyECL\
```

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

ecl run	executes the supplied ecl run command
<b>Arguments</b>	
target	The target cluster to which to publish
wuid	The workunit id to run (WUID is case-sensitive)
file	The ECL text file to run
archive	The ECL archive to run
so   dll	The workunit dynamic linked library or shared object to run
-	Specifies object should be read from stdin
<b>Options</b>	
--job-name	The job name to use for the workunit
-in,--input=<file xml>	The file or xml content to use as query input
-X<name>=<value>	Sets the stored input value (stored('name'))
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
--poll	Submits a job asynchronously and polls the server until the state of the workunit changes to completed. It then retrieves the results. Combine with the --wait option to limit the time that it polls.
--exception-level	Sets the minimum severity for reporting exceptions. Possible severity levels are <b>info</b> , <b>warning</b> , or <b>error</b> . The default is info which returns all exceptions.
--protect	This option prevents the workunit from deletion
<b>ECL Options</b>	
--main=<definition>	This option allows you to compile queries from existing source trees or repositories. It can be used to specify default repositories and versions to use. See the "Git Support" section for specific usage.

HPCC Systems® Client Tools  
ECL Command Line Interface

--snapshot, -sn=<label>	Snapshot label to use from a legacy ECL repository
--ecl-only	Send ECL query to HPCC Systems cluster as text rather than as a generated archive
--limit=<limit>	Sets the result limit for the query
-f<option>[=value]	Set an ECL option (equivalent to #option)
-f-<option>[=value]	Set an eclcc command line option with the single dash ('-')
-f--<option>[=value]	Set an eclcc command line option with the double dash ('--')
-Dname=value	Override the definition of a global attribute 'name'
<b>eclcc Options</b>	
-I <path>	Add path to locations to search for ecl imports
-L <path>	Add path to locations to search for system libraries
--manifest	Specify path to the manifest file
-g, --debug	Enable debug symbols in generated code
-checkDirty	This option reports any modified attributes using git status
--fetchrepos	This option automatically downloads any missing repositories associated with dependencies
-R<repo>[#version]=<path>	This option resolves repository references specified in directory <path>
--updaterepos	Setting this option automatically updates repositories associated with dependencies
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl results

**ecl results <wuid> [--noroot] [--exception-level=<value>]**

### Examples:

```
ecl results W20170519-142920
ecl results W20170519-142920 --noroot --exception-level=error
```

ecl results	returns the full results of a given WUID in XML format.
<b>Arguments</b>	
wuid	The workunit from which to return results. (WUID is case-sensitive)
<b>Options</b>	
--noroot	Suppresses the <Result> root tag in the XML returned.
--exception-level	Sets the minimum severity for reporting exceptions. Possible severity levels are <b>info</b> , <b>warning</b> , or <b>error</b> . The default is info which returns all exceptions.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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.
<b>Arguments</b>	
target	The name of target queryset containing query to activate
query_id	The query to activate
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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.
<b>Arguments</b>	
target	The name of the target queryset containing the alias to deactivate
active_alias	The active alias to be removed from the queryset
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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.
<b>Actions</b>	
list	List queries in queryset(s)
<b>Options</b>	
<target>	The name of queryset from which to list queries
-t, --target	The name of the target cluster associated with the queries to list
--inactivate	Show only the queries that do not have an active alias
--show=<flags>	Show only queries with matching flags
<b>Flags</b>	
A	Active
S	Suspended
U	No Flags
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)



## ecl queries files

### ecl queries files <target> [<query>]

The queries files command displays a list of the files currently in use by the given query. If the *query* option is omitted, it returns a list of files for all queries on the specified target.

Examples:

```
ecl queries files roxie myquery
```

Example result:

```
> ecl queries files roxie myquery
-----
Query: myquery
Files used:
  jd::subfile1, 100 bytes, 2 part(s)
  jd::subfile2, 100 bytes, 2 part(s)

SuperFiles used:
  jd::mysuperfile
> jd::subfile2
> jd::subfile1
```

Options	
target	Name of target cluster on which the query is published
query	Optional. Name of the query for which to get a list of files it uses. If omitted, a list of files for each query on the specified target is returned.
Common Options	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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 Systems 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.
<b>Actions</b>	
copy	Copy a query from one queryset to another
<b>Options</b>	
source_query_path	The path of the query to copy using the format: [//ip:port/]queryset/query or queryset/query.
<target>	The name of the target cluster to copy the query to
--source-ssl	Use SSL when connecting to the source (default if --ssl is enabled)
--source-no-ssl	Do not use SSL when connecting to source (default if --ssl is NOT enabled)
--no-files	Specifies to not copy DFS file information for files referenced by the query
--daliip=<IP>	IP address or hostname of the remote Dali DFS to use for copying file information. Only required when remote environment is less than version 3.8
--source-process	Process cluster to copy files from
-A, --activate	Activates the query when copied
-sp, --suspend-prev	Suspend previously active query
-dp, --delete-prev	Delete previously active query
--no-reload	Specifies to not request a reload of the Roxie cluster
-O, --overwrite	Whether to overwrite existing information - true if present
--update-super-files	Update local DFS super-files if remote DALI has changed
--update-clone-from	Update local clone from location if remote DALI has changed
--dont-append-cluster	Only use to avoid locking issues due to adding cluster to file
--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. Prevents failure if using foreign files in Roxie query.
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
--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.
--priority=<val>	Set the priority for the query. Values can be LOW, HIGH, SLA, NONE. NONE will clear the current setting.

HPCC Systems® Client Tools  
ECL Command Line Interface

--comment=<string>	Set a comment associated with the query
-n, --name=<value>	The destination name for the copied query
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl queries copy-set

**ecl queries copy-set <source\_target> <destination\_target> [--all] [--clone-active-state]**

Examples:

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

ecl queries copy-set	Copies a set of queries from one target to another.
<b>Actions</b>	
copy-set	Copy a set of queries from one target to another.
<b>Options</b>	
source_target	Target cluster from which to copy queries.
destination_target	Target cluster to copy queries to.
--source-ssl	Use SSL when connecting to the source (default if --ssl is enabled)
--source-no-ssl	Do not use SSL when connecting to source (default if --ssl is NOT enabled)
--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.
-O, --overwrite	Whether to overwrite existing DFS information - true if present
--update-super-files	Update local DFS super-files if remote DALI has changed
--update-clone-from	Update local clone from location if remote DALI has changed
--dont-append-cluster	Only use to avoid locking issues due to adding cluster to file
--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.
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	

HPCC Systems® Client Tools  
ECL Command Line Interface

--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl queries config

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

Examples:

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

ecl queries config	Updates query configuration values
<b>Actions</b>	
config	Set or update query configuration values
<b>Options</b>	
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=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
--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.
--priority=<val>	Set the priority for the query. Values can be LOW, HIGH, SLA, NONE. NONE will clear the current setting.
--comment=<string>	Set a comment associated with the query
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl queries recreate

**ecl queries recreate <target> <query> [<destination-target>] [options]**

Examples:

```
ecl queries recreate roxie findpeople
ecl queries recreate roxie findpeople roxie2
```

ecl queries recreate	Recompiles a query into a new workunit and republishes the new workunit. This is useful when upgrading to a new ECL compiler and you want to recompile a query from the exact same source. The ECL archive must be available within the workunit of the query.
<b>Actions</b>	
recreate	Recompiles a query into a new workunit and republishes the new workunit.
<b>Arguments</b>	
target	The target the query you wish to recreate is in
query	The query ID of the query you wish to recreate
destination-target	Optional: The target you want to move the new query to (if different from the source target)
<b>Options</b>	
-A, --activate	Activates query when published
--limit=<limit>	Sets the result limit for the query, defaults to 100
-sp, --suspend-prev	Suspends previously active query
-dp, --delete-prev	Deletes previously active query
-A-, --no-activate	Does not activate query when published
--no-publish	Creates a recompiled workunit, but does not publish it
--no-reload	Specifies to not request a reload of the Roxie cluster
--no-files	Specifies to not copy DFS file information for files referenced by the query
--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.
--daliip=<IP>	IP address or hostname of the remote Dali to use for remote logical file lookups.
--update-super-files	Update local DFS superfiles if remote DALI has changed
--update-clone-from	Update local clone from location if remote DALI has changed
--dont-append-cluster	Use only to avoid locking issues due to adding cluster to file
--source-process=<value>	Process cluster to copy files from
--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.
--priority=<val>	The priority for this query. Value can be LOW, HIGH, SLA, NONE. NONE will clear current setting.
--comment=<string>	A comment associated with this query

HPCC Systems® Client Tools  
ECL Command Line Interface

--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)



## ecl queries import

**ecl queries import <target> <file> [--clone-active-state] [--replace] [options]**

Example:

```
ecl queries import roxie1 myqueryset.xml
```

ecl queries import	Imports the contents of a queryset previously exported to disk
<b>Actions</b>	
import	Imports a queryset from a file
<b>Arguments</b>	
target	The target cluster to import queries to
<b>Options</b>	
--all	Copy both active and inactive queries. If omitted, only active queries are imported.
--replace	Replace entire existing queryset
--queries	Filter query ids to select for import
--no-files	Specifies to not copy DFS file information for files referenced by the query
--daliip=<IP>	Remote Dali DFS to use for copying file information
--source-process=<value>	Process cluster to copy files from
--clone-active-state	Make copied queries active if active on source
-O, --overwrite	Completely replace existing DFS file information (dangerous)
--update-super-files	Update local DFS superfiles if remote DALI has changed
--update-clone-from	Update local clone from location if remote DALI has changed
--dont-append-cluster	Use only to avoid locking issues due to adding cluster to file
--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.
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	
--help	Displays usage information for the given command

HPCC Systems® Client Tools  
ECL Command Line Interface

-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl queries export

**ecl queries export <target> [options]**

Example:

```
ecl queries export roxie1 --output=myqueryset.xml
```

ecl queries export	saves backup information about a given queryset to a file.
<b>Actions</b>	
export	Exports queryset information to a file
<b>Arguments</b>	
target	Name of target cluster to export from
-O, --output=<filename>	Filename to save exported backup information to (optional)
<b>Options</b>	
--active-only	Only include active queries in the exported queryset.
--protect	Protect the workunits for the included queries
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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
<b>Actions</b>	
add	Adds a package map to the target cluster
<b>Arguments</b>	
target	The target to associate the package map with
filename	The name of the file containing package map information.
--daliip=	IP address or hostname of the remote Dali to use for logical file lookups
<b>Options</b>	
-O, --overwrite	Whether to overwrite existing information - true if present
-A, --activate	Activates package map
--allow-foreign	Specifies to allow the use of foreign files. If a package map references foreign files and this is not enabled, package map add will fail.
--pmid=<packagemapid>	id of package map - defaults to filename if not specified
--global-scope	The packagemap specific with this option can be shared across multiple targets
--preload-all	Sets the preload files option for all packages
--replace	Replace the existing packagemap
--update-superfiles	Updates the local DFS super-files if the remote DALI has changed
--update-clone-from	Update the local clone from location if the remote DALI has changed
--dont-append-cluster	Only use to prevent locking issues due to adding cluster to file
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	

HPCC Systems® Client Tools  
ECL Command Line Interface

--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl packagemap delete

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

Examples:

```
ecl packagemap delete roxie mypackagemap
```

ecl packagemap delete	Calls the packagemap delete command
<b>Actions</b>	
delete	Deletes a package map
<b>Options</b>	
<target>	The name of the target to use when deleting the packagemap
<filename>	The name of the packagemap to delete
--global-scope	Use when the specified packagemap is shared across multiple targets
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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 package map and make the specified package map active.
<b>Arguments</b>	
target	The target containing the package map to activate
packagemap	name of package map to update
--global-scope	The specified packagemap can shared across multiple targets
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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 package map.
<b>Arguments</b>	
target	The target containing the package map to deactivate
packagemap	Name of package map to deactivate
--global-scope	The specified packagemap can shared across multiple targets
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)



## ecl packagemap list

**ecl packagemap list <target>**

Examples:

```
ecl packagemap list roxie
```

ecl packagemap list	Calls the packagemap list command
<b>Actions</b>	
list	Lists loaded package map names
<b>Arguments</b>	
target	The target containing the package map to list
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl packagemap info

**ecl packagemap info [options] <target>**

Examples:

```
ecl packagemap info roxie
```

ecl packagemap info	Calls the packagemap info command
<b>Actions</b>	
info	returns package map info
<b>Arguments</b>	
target	The target containing the package map to retrieve
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl packagemap add-part

**ecl packagemap add-part <target> <pmid> <filename>**

Examples:

```
ecl packagemap add-part roxie multipart.pkg addresses.pkg
```

The packagemap add-part command adds additional package map content to an existing package map

ecl packagemap add-part	Calls the packagemap add-part command.
<b>Actions</b>	
add-part	Adds additional package map content to an existing package map
<b>Arguments</b>	
target	Name of target to use when adding package map part
pmid	Identifier of package map to add the part to
filename	one or more part files
<b>Options</b>	
--part-name	Name of part being added (defaults to filename)
--delete-prev	Replace an existing part with matching name
--daliip=<ip>	IP of the remote Dali to use for logical file lookups
--global-scope	The specified package map is shared across multiple targets
--source-process=<value>	Process cluster to copy files from
--allow-foreign	Do not fail if foreign files are used in package map
--preload-all	Set preload files option for all packages
--update-super-files	Update local DFS superfiles if remote DALI has changed
--update-clone-from	Update local clone from location if remote DALI has changed
--dont-append-cluster	Use only to avoid locking issues due to adding cluster to file
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	
--help	Displays usage information for the given command

HPCC Systems® Client Tools  
ECL Command Line Interface

-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl packagemap get-part

**ecl packagemap get-part <target> <packagemap> <partname>**

Examples:

```
ecl packagemap get-part roxie multipart.pkg contacts
```

The get-part command fetches the given part from the given package map

ecl packagemap get-part	Calls the packagemap get-part command.
<b>Actions</b>	
get-part	Fetches the given part from the given package map
<b>Arguments</b>	
target	Name of target to use when adding package map part
packagemap	Name of the package map containing the part
partname	Name of the part to retrieve
<b>Options</b>	
--global-scope	The specified package map is sharable across multiple targets
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl packagemap remove-part

**ecl packagemap remove-part <target> <pmid> <partname>**

Examples:

```
ecl packagemap remove-part roxie multipart.pkg contacts
```

The remove-part command will remove the given part from the given package map

ecl packagemap remove-part	Calls the packagemap remove-part command.
<b>Actions</b>	
remove-part	Removes the given part from the given package map
<b>Arguments</b>	
target	Name of target to use
packagemap	Name of the package map containing the part
partname	Name of the part to remove
<b>Options</b>	
--global-scope	The specified package map is sharable across multiple targets
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl packagemap validate

**ecl packagemap validate <target> [<filename>]**

Examples:

```
ecl packagemap validate roxie mypackagemap.pkg
ecl packagemap validate roxie --active
ecl packagemap validate roxie mypackagemap.pkg --ignore-queries='findPerson*, findByZip'
```

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 package map.

Filename, --active, and --pmid are mutually exclusive. The --active or --pmid options validate a package map 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.
<b>Actions</b>	
validate	Validates package map info
<b>Arguments</b>	
filename	The filename containing the package map info to validate
target	The target containing the package map to validate
--active	Validates the package map that is active for the given target
--check-dfs	This option verifies that subfiles exist in the DFS
-pm, --pmid	Validates the specified packagemap
--queryid	Validate the files for the given queryid if they are mapped in the package map
--ignore-optional	Will not report optional files that are not defined in the packagemap
--ignore-warnings	Will not report general packagemap warnings
--ignore-queries=<setOfQueries>	Will not report on the queries which match the expression. The set of queries can be a comma-separated list or you can use the option once per entry. Wildcards are supported.
--global-scope	The specified package map is sharable across multiple targets
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.

HPCC Systems® Client Tools  
ECL Command Line Interface

--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)



## ecl packagemap copy

**ecl packagemap copy <path> <target>**

Copies a package map from one target to another.

Examples:

```
ecl packagemap copy roxie/MyPkg roxie2
ecl packagemap copy //192.168.0.100:8010/roxie/MyPkg roxie2
```

ecl packagemap copy	Calls the packagemap copy command
<b>Actions</b>	
copy	Copies a package map from one target to another
<b>Arguments</b>	
path	Path to the source package map to copy.
target	The target to copy the package map to
<b>Options</b>	
-A, --activate	Activates package map
--daliip=	IP address or hostname of the remote Dali to use for logical file lookups
--pmid=<packagemapid>	id of package map - defaults to filename if not specified
--source-process	Process cluster to copy files from
--preload-all	Set preload files option for all packages
--replace	Replace existing packagmap
--update-super-files	Update local DFS superfiles if remote Dali has changed
--update-clone-from	Update local clone from location if remote Dali has changed
--dont-append-cluster	Only use to avoid locking issues due to adding cluster to file
<b>DFU Options</b>	
--dfu-copy	Use DFU to copy files during deployment, not on Roxie in the background
--init-publisher-wuid	Preallocates the publisher workunit at the beginning of the command and displays the WUID on the command line
--dfu-queue	The DFU Queue to use when doing a DFU copy
--dfu-wait	The amount of time in seconds to wait for the DFU copy to complete (if neither --only-copy-files or --stop-if-files-copied are specified). Default is 1800 (30 minutes)
--dfu-overwrite	Specifies that DFU copy command should overwrite physical files that are already on disk.
--only-copy-files	Copy the files needed for the query, but don't publish the query
--stop-if-files-copied	If all files already exist, publish the query. Otherwise, copy the files needed for the query, but don't publish the query
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

For the path, the following formats are supported:

- remote packagemap: //IP:PORT/Target/PackageMapId
- local packagemap: target/PackageMapId

## ecl roxie attach

**ecl roxie attach <processName>**

Examples:

```
ecl roxie attach myroxie
```

ecl roxie attach	Attach the roxie to Dali
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl roxie detach

**ecl roxie detach <processName>**

Examples:

```
ecl roxie detach myroxie
```

ecl roxie detach	Detach the roxie from Dali
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl roxie reload

**ecl roxie reload <processName>**

Examples:

```
ecl roxie reload myroxie
```

ecl roxie reload	Reloads the roxie info from Dali
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl roxie check

**ecl roxie check <processName>**

Examples:

```
ecl roxie check myroxie
```

ecl roxie check	Checks the state of the roxie process
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl roxie xref

### ecl roxie xref <cluster>

The **roxie xref** command returns file information for the specified queries on the specified cluster. If the *queryids* option is omitted, file information about all queries is returned. The result is in XML format.

Examples:

```
ecl roxie xref myroxie
ecl roxie xref myroxie --queryids=myquery.1,myotherquery.1
```

Example result:

```
<QueryXrefInfo>
  <Endpoint ep="192.168.56.1:9876">
    <Queries reporting="1">
      <Query id="myquery.1"/>
        <SuperFile name="jd:mysuperfile">
          <File name="jd:subfile1"/>
          <File name="jd:subfile2"/>
        </SuperFile>
      <Query id="myotherquery.1"/>
        <SuperFile name="jd:myothersuperfile">
          <File name="jd:subfile1"/>
          <File name="jd:subfile2"/>
          <File name="jd:subfile3"/>
        </SuperFile>
      </Queries>
    </Endpoint>
  </QueryXrefInfo>
```

ecl roxie xref	Returns file information for the selected queries in XML format.
<b>Options</b>	
--check-all-nodes	Gets query file information from all nodes. This can be slow.
--queryids=<csv list>	The queries for which to get file information (default is all queries)
--wait=<ms>	Maximum time (in milliseconds) to wait for cluster finish updating. Any value lower than the minimum of 1000 is ignored.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)

<code>--wait-read=&lt;Secs&gt;</code>	Timeout while reading from socket (in seconds)
---------------------------------------	--



## 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
<b>Options</b>	
<bundleName>	The name of a bundle file or installed bundle
--recurse	Displays indirect dependencies
--version	Specify a version of the bundle
--help	Displays usage information about the given command
-v, --verbose	Output additional tracing information

## ecl bundle info

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

Examples:

```
ecl bundle info mybundle  
ecl bundle info https://github.com/hpcc-systems/ecl-bundles.git  
ecl bundle info mybundle --version=2
```

ecl bundle info	Lists information about a bundle
<b>Options</b>	
<bundle>	A bundle filename, a bundle folder, a bundle name, or a URL.
--branch	Names a branch to install when bundle references a git repository
--remote	Interprets the bundle name as a remote name from catalog
--version	Specify a version of the bundle
--help	Displays usage information about the given command
-v, --verbose	Output additional tracing information

If a URL ends in .git, it is assumed to be a git repository (fetched using git clone) otherwise it is assumed to be the URL of a file that can be retrieved. In either case, it is fetched to a temporary local location, processed as a local file/directory and then removed.

## ecl bundle install

**ecl bundle install <bundleName>**

Examples:

```
ecl bundle install mybundle
ecl bundle install https://github.com/hpcc-systems/ecl-bundles.git
ecl bundle install mybundle --dryrun
ecl bundle install mybundle --update
ecl bundle install mybundle --keeprior
```

ecl bundle install	Installs a bundle
<b>Options</b>	
<bundleName>	The name or URL of a bundle file, folder, or installed bundle.
--dryrun	List what would be installed, but do not copy
--force	Install even if required dependencies missing
--keeprior	Do not remove any previous versions of the bundle
--remote	Interprets the bundle name as a remote name from catalog
--update	Update an existing installed bundle
--help	Displays usage information about the given command
-v, --verbose	Output additional tracing information

If a URL ends in .git, it is assumed to be a git repository (fetched using git clone) otherwise it is assumed to be the URL of a file that can be retrieved. In either case, it is fetched to a temporary local location, processed as a local file/directory and then removed.

To use the "ecl bundle install <git url>" command, you must have git installed and configured on your system. Git must be accessible to the user (in the path).

### Server-side Installation

A system using ECL Server and a remote MySQL repository or ECLCC Server with git hooks can only access bundles if they are installed on the server. Using a bundle in the ECL Playground also requires installation on the ECLCC Server node.

1. In a terminal window on your ECLCC Server (or ECL Server), use this command:

```
sudo su hpcc
```

This switches the user to **hpcc**.

2. Next install the bundle using this command:

```
ecl bundle install <bundle URL>.git
```

For example:

```
ecl bundle install https://github.com/hpcc-systems/Visualizer.git
```

3. Close the terminal window or use the **exit** command to return to acting as the original user.

## 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
<b>Options</b>	
<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
--help	Displays usage information about the given command
-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
<b>Options</b>	
<pattern>	A pattern specifying bundles to list. If omitted, all bundles are listed
--details	Report details of each installed bundle
--help	Displays usage information about the given command
-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
<b>Options</b>	
<bundle>	The name of a bundle file
--version	The version of the bundle to make active, or "none"
--help	Displays usage information about the given command
-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.process_cluster
process_cluster	The roxie process cluster to check, or
--roxies <list>	A comma-separated list of roxies to check
--planes <list>	A comma-separated list of data planes to search for files
--check-packagemaps	Exclude files referenced in active package maps
--delete	Deletes unused files from DFS
--delete-subfiles	Deletes unused files from DFS and removes them from superfiles.
--delete-recursive	Deletes unused files from DFS and removes them from superfiles recursively.
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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
<b>Options</b>	
-wu	The WUID (Workunit ID) (WUID is case-sensitive)
-n	The job name
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)



## 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.
<b>Options</b>	
-wu	The WUID (Workunit ID) (WUID is case-sensitive)
-n	The job name
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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.
<b>Options</b>	
-n	The job name
--limit= <i>nn</i>	Integer to set result limit, default is 100
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be promoted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## 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) (WUID is case-sensitive)
<b>Options</b>	
--limit=<limit>	This sets the result limit. This is useful when using wildcards in a request. (Default is 100)
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl zapgen

**ecl zapgen <WUID> --path <zap\_file\_path> [options]**

Use the zapgen command to create a Zipped Analysis Package (Z.A.P.) containing collecting system information about a workunit and encapsulating it into a shareable package. It is a useful tool for reporting errors, inconsistencies, or other unexpected behavior.

Examples:

```
ecl zapgen W20171017-091320 --path ~/reports
ecl zapgen W20171018-091399 --path ~ --inc-thor-slave-logs --description "Unexpected result from JOIN"
```

ecl zapgen	Generates and downloads a ZAP file for the given workunit in the specified path.
WUID	The WUID (Workunit ID) (WUID is case-sensitive)
--path	The path to store the ZAP file
<b>Options</b>	
--inc-thor-slave-logs	Includes Thor slave(s) log into the ZAP file
--description	Description of the issue
--create-dirs	Creates the necessary directory tree
<b>Common Options</b>	
--help	Displays usage information for the given command
-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.
--accept-self-signed	Allows SSL servers to use self signed certificates
--cert	Path to file containing SSL client certificate
--key	Path to file containing SSL client certificate private key
--cacert	Path to file containing SSL CA certificate
--port	The ECL services port (Default is 8010)
-u, --username	The username (if necessary)
-pw, --password	The password (if necessary). If you supply a username and do not supply a password, you will be prompted for a password.
--wait-connect=<Ms>	Timeout while connecting to server (in milliseconds)
--wait-read=<Secs>	Timeout while reading from socket (in seconds)

## ecl url-secret-name

**ecl url-secret-name url** [--username=<username>]

ecl url-secret-name	Generates a secret name from a URL for automatic URL mapping.
url	The URL to convert into a secret name.
--username	Optional. The username to associate with the URL. This will override any user-name embedded in the URL.

The *ecl url-secret-name* command generates a secret name from a URL that can be used to support ECL SOAPCALL/HTTPCALL automated URL to Secret mapping.

A username can either be embedded in the URL, such as `https://username@example.com`, or passed in as a parameter using the `--username=username` option. If a username is passed in as a parameter it overrides a username in the URL.

Passwords embedded in the URL are not needed and will be ignored.

When ECL SOAPCALL URL secret mapping is enabled SOAPCALL will convert the URL provided into a name of this format. ECL will then attempt to lookup the secret, and if found, will use the contents of the secret, rather than the original URL.

Examples:

```
ecl url-secret-name https://example.com --username jimi  
ecl url-secret-name http://example.com --username jimi
```

## ecl roxie memlock

**ecl roxie memlock <process\_cluster>**

<i>Options</i>	
<i>process_cluster</i>	The Roxie process cluster to memlock

The *ecl roxie memlock* command locks Roxie heap memory. This helps to prevent intermittent slow queries by ensuring that the operating system does not swap the memory out.

Examples:

```
ecl roxie memlock mycluster
```

## ecl roxie memunlock

**ecl roxie memunlock <process\_cluster>**

<i>Options</i>	
<i>process_cluster</i>	The Roxie process cluster to memunlock

The *ecl roxie memunlock* command unlocks Roxie heap memory.

Examples:

```
ecl roxie memunlock mycluster
```

## ecl roxie getmemlocked

**ecl roxie getmemlocked <process\_cluster>**

<i>Options</i>	
<i>process_cluster</i>	The Roxie process cluster to getmemlocked status

The *ecl roxie getmemlocked* command retrieves the Roxie heap memory lock status.

Examples:

```
ecl roxie getmemlocked mycluster
```



# ECL Compiler

The ECL Compiler is the compiler component of the High Performance Computing Cluster (HPCC) Systems platform. It is embedded and included when you install the HPCC Systems platform. 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, 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 <i>ECL Language Reference</i> for details.
-main <ref>	Compile definition <ref> from the specified source - note extended Git repository support (see "Git Support" in the previous section)
-syntax	Perform a syntax check of the ECL
-platform=hthor	Generate code for hthor cluster
-platform=roxie	Generate code for roxie cluster (default)
-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
-qa	Save ECL query archive 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, --debug	Enable debug symbols in generated code
-Wc,xx	Pass option xx to the C++ compiler
-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)
-save-temps	Do not delete intermediate files
-shared	Generate workunit shared object instead of a stand-alone executable

File resolution options:

-dfs=ip	Use specified ip for DFS filename resolution
-scope=prefix	Use specified scope prefix in DFS filename resolution
-user=id	Use specified username in DFS filename resolution
-password=xxx	Use specified password in DFS filename resolution (blank to prompt)

Other options (list is available using eclcc -help -v):

-aoption[=value]	Set an application option
--allow=str	<p>Allow use of named feature. (e.g., cpp, pipe, all)</p> <p><b>cpp</b>: Allow embedded code within ECL (e.g., C++, JAVA, Python, etc.)</p> <p><b>pipe</b>: Allow the PIPE command to send data to an external program.</p> <p><b>userECL</b>: Allow code that is not found via the ECL include paths</p> <p><b>datafile</b>: Allow access to datafiles from ECL.</p> <p><b>extern</b>: Allow access to an external service function</p> <p><b>all</b>: Allow all features</p>
-allowsigned	Only allows access to a feature from signed code
-fisComplexCompile	Prevents attempts to compile as a child process when a query is complex.
-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
-checkDirty	Causes eclcc to generate a warning for any attribute that has been modified (according to the output of git status). Use of this function requires that git be installed and available on the path.
--component	Set the name of the component this is executing on behalf of
-Dname=value	Override the definition of a global attribute 'name'
--deny=all	Disallow use of all named features not specifically allowed using --allow
--deny=str	<p>Disallow use of named feature</p> <p><b>cpp</b>: Disallow embedded code within ECL (e.g., C++, JAVA, Python, etc.)</p> <p><b>pipe</b>: Disallow the PIPE command to send data to an external program.</p>
--expand <path>	Expand the contents of an archive to the specified directory. The contents of the submitted query will output to stdout.
--fastsyntax	Delay expanding functions when parsing. May speed up processing for some queries
-help, --help	Display help message
--help -v	Display verbose help message
--internal	Run internal tests
--legacy	Use legacy import semantics (deprecated)

Other options (continued):

--leakcheck	Clean up memory since checking for memory leaks
--keywords	Outputs the lists of ECL reserved words to stdout (XML format)
-legacyimport	Use legacy import semantics (deprecated)
-legacywhen	Use legacy when/side-effects semantics (deprecated)
--logfile <file>	Write log to specified file
--logdetail= <i>n</i>	Set the level of detail in the log file
--maxErrors=<n>	Limit the number of errors, aborting on the nth (default = 5)
--metacache= <i>x</i>	Specify directory to store distributed meta information from the eclcc indexer. To disable the indexer, set to an empty value using "--metacache=". If omitted, the default location is .eclcc/metacache.
--nologfile	Do not write any log file
--nogpg	Do not run gpg to check signatures on signed code
--nosourcepath	Compile as if the source came from stdin
--nostdinc	Do not include the current directory in -I
-pch	Generate precompiled header for eclinclude4.hpp
-P <path>	Specify the path of the output files (only with -b option)
-showpaths	Print information about the search paths eclcc is using
-specs <file>	Read eclcc configuration from specified file
-split <i>m:n</i>	Process a subset <i>m</i> of <i>n</i> input files (only with -b option)
-v --verbose	Output additional tracing information while compiling
-wxxx=level	Set the severity for a particular warning code or category.  Valid options for level are: all   ignore   log   warning   error   fail <b>-wall</b> sets default severity for all warnings
--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*

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

<i>server</i>	The URL (http:// or https://) and/or IP address of the ESP server. The port may also be included.
<i>username</i>	A userid with authorized access to the <i>server</i> .
<i>password</i>	The password authorizing access for the <i>username</i> .
<i>overwrite</i>	Optional. A boolean flag (0   1) indicating whether to overwrite any existing file of the same name. If omitted, the default is 0.
<i>replicate</i>	Optional. A boolean flag (1   0) indicating whether to replicate the file. If omitted, the default is 1.

	<b>This option is only available on systems where replication has been enabled.</b>
<i>autorecover</i>	Optional. The number of times to attempt recovery of a failed <i>operation</i> . If omitted, the default is 0.
<i>nowait</i>	Optional. A boolean flag (0   1) indicating whether to return immediately without waiting for completion of the <i>operation</i> . If omitted, the default is 0.
<i>connect</i>	Optional. The number of simultaneous connections to limit the <i>operation</i> to. If omitted, the default is 25.
<i>nocommon</i>	Optional. A boolean flag (0   1, default=1). Set to 0 to enable "commoning up" of pull or push processes on same host.
<i>throttle</i>	Optional. The transfer speed (in Mbits/second) to restrict the <i>operation</i> 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.
<i>norecover</i>	Optional. A boolean flag (0   1) indicating whether to create or recover the <i>operation</i> from recovery information. If omitted, the default is 0.
<i>nosplit</i>	Optional. A boolean flag (0   1) indicating whether to split file parts to multiple target parts. If omitted, the default is 0.
<i>compress</i>	Optional. A boolean flag (0   1) indicating whether to compress the target file.
<i>push</i>	Optional. A boolean flag (0   1) indicating whether to override push/pull default.
<i>encrypt=&lt;password&gt;</i>	Optional. Specifies to encrypt the target filename using the supplied password.
<i>decrypt=&lt;password&gt;</i>	Optional. Specifies to decrypt the source filename using the supplied password.
<i>jobname=&lt;jobname&gt;</i>	Specify a jobname for the DFU operation's workunit.
<i>transferbuffersize=nnn</i>	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:

```
;The values below are examples, you should change them to match your platform deployment
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 Systems cluster.

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 <i>srcxml</i> or <i>srcplane</i> parameter.
srcplane	Optional. The source storage plane containing the source file. Note: <i>srcplane</i> should not be used at the same time as <i>srcip</i> .
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 <i>dstname</i> . If omitted, the information must be supplied by the <i>srcxml</i> parameter.
srcxml	The name of the XML file containing the information required for the <i>srcip</i> and <i>srcfile</i> parameters. This file may have been obtained by previous use of the <i>savexml</i> operation. This option provides the feature of combining multiple source files into a single resulting logical file in the HPCC Systems cluster.
dstname	The logical name of the destination file.
dstcluster	The name of the destination cluster. For a containerized deployment, set this to the target data plane (For example, dstcluster=data)
prefix	Optional. Both of the following (separated by a comma):
filename{:length}	Prepends the filename (optionally limited to <i>length</i> 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>B</b> specifies big endian, <b>L</b> specifies little endian) and the size of integer to contain it ( <b>1</b> to <b>8</b> bytes). If format and size are omitted, the default is L4.  When using wildcard characters (* and ?) to spray multiple source files (srcfile) to a single dstname, you <b>MUST</b> use both the filename and filesize options if you need to be able to despray the contents of each record in the dstname back to the multiple source files they originally came from. If you never need to do that, then the filesize option may be omitted.

<i>expireDays</i>	Optional. Specifies the file is a temporary file to be automatically deleted after the specified number of days since the file was read. If omitted, the default is -1 (never expires). If set to 0, the file is automatically deleted when it reaches the threshold set in Sasha Server's <b>expiryDefault</b> setting.
<i>format</i>	Optional. One of the following values: <b>fixed csv delimited xml recfmv recfmb</b> . If omitted, the default is fixed.
<b>fixed</b> format options:	
recordsize	The fixed size of each record, in bytes.
<b>csv/delimited</b> 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 (').
<b>xml</b> format options:	
rowtag	The XML tag identifying each record. Required.
encoding	Optional. One of the following: utf8 utf8n utf16 utf16le utf16be utf32 utf32le utf32belf 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=/var/lib/HPCCSystems/mydropzone/timezones.txt dstname=RTTEMP::timezones.txt
    dstcluster=mythor format=fixed recordsize=155

//fixed spray example using a srcxml file:
dfuplus action=spray srcxml=/var/lib/HPCCSystems/mydropzone/flattimezones.xml
    dstname=RTTEMP::timezones.txt dstcluster=mythor recordsize=155

//csv spray example:
dfuplus action=spray srcip=10.150.50.14
    srcfile=/var/lib/HPCCSystems/mydropzone/timezones.csv dstname=RTTEMP::timezones.csv
    dstcluster=mythor format=csv

//the spray.xml file contains:
<File directory="/var/lib/HPCCSystems/mydropzone/"
    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=mythor dstname=RTTEMP::myzip1 recordsize=5

//xml spray example:
dfuplus action=spray srcip=10.150.50.14
    srcfile=/var/lib/HPCCSystems/mydropzone/timezones.xml dstname=RTTEMP::timezones.xml
    dstcluster=mythor format=xml rowtag=area

//Multiple spray all .JPG and .BMP files under
// /var/lib/HPCCSystems/mydropzone/ on 10.150.51.26 to single logical file LE::imagedb
dfuplus action=spray srcip=10.150.51.26
    srcfile=/var/lib/HPCCSystems/mydropzone/*.jpg,/var/lib/HPCCSystems/mydropzone/*.bmp
    dstcluster=mythor
    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;
// using srcplane
dfuplus action=spray srcplane=lzstorageplane
    srcfile=/var/lib/HPCCSystems/dropzone/largedatal
    dstname=mytest::test:spraytest
    dstcluster=mydatastorageplane recordsize=58 overwrite=1 server=127.0.0.1:8010

```

## 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:

<i>srcname</i>	The logical name of the source file. This may contain wildcard characters (*) and ?) to include multiple source files in the despray to a single <i>dstfile</i> .
<i>dstip</i>	Optional. The IP address of the destination machine. If omitted, the information must be supplied by the <i>dstxml</i> or <i>dstplane</i> parameter. Deprecated, you should use <i>dstplane</i> instead.
<i>dstplane</i>	Optional. The destination storage plane. Note: <i>dstplane</i> should not be used at the same time as <i>dstip</i> . In a containerized deployment, <i>dstplane</i> is required if you have more than one Landing Zone.
<i>dstfile</i>	Optional. The path to the destination file. This may contain wildcard characters (*) and ?) to despray a single <i>srcname</i> to multiple <i>dstfiles</i> . If omitted, the information must be supplied by the <i>dstxml</i> parameter.
<i>dstxml</i>	The name of the XML file containing the information required for the <i>dstip</i> and <i>dstfile</i> parameters. This file may have been obtained by previous use of the

	savexml <i>operation</i> . This option provides the feature of splitting a single resulting logical file in the cluster into multiple destination files.
<i>splitprefix</i>	Optional. Both of the following (separated by a comma):
<b>filename{:<i>length</i>}</b>	Uses the prepended filename (see the <i>prefix</i> option to the <i>spray operation</i> ) to split out the data into separate files.
<b>filesize{:[B L][1-8]}</b>	Uses the prepended size of the file (see the <i>prefix</i> option to the <i>spray operation</i> ) to split out the data into separate files.  When using wildcard characters (* and ?) to spray multiple source files (srcfile) to a single dstname, you MUST use both the filename and filesize options if you need to be able to despray the contents of each record in the dstname back to the multiple source files they originally came from. If you never need to do that, then the filesize option may be omitted.
<i>wrap= 0   1</i>	Optional. If set to 1, desprays as multiple files on the landing zone. Default is 0.
<i>multicopy= 0   1</i>	Optional. If set to 1, each destination part gets the whole file. Default is 0.

#### Examples:

```
dfuplus action=despray dstip=10.150.50.14
  dstfile=/var/lib/HPCCSystems/mydropzone/timezones.txt srcname=RTTEMP::timezones.txt
//the spray.xml file contains:
<File directory="/var/lib/HPCCSystems/mydropzone/"
  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=mythor
  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=/var/lib/HPCCSystems/mydropzone/
  splitprefix=FILENAME,FILESIZE

// using dstplane
```

```
dfuplus action=spray srcname=mytest::test:spraytest
dstplane=lzstorageplane
dstfile=mydespraytest server=127.0.0.1:8010
```

## 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 Systems 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.

To copy a superfile and retain its structure, use the *copysuper* operation. If you use the copy operation for a superfile, it consolidates all the superfile content into a single logical file on the target, not a superfile. This is not valid for a superfile containing INDEXes and will produce an error.

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

<i>srcname</i>	The logical name of the source file.
<i>dstname</i>	The logical name of the destination file.
<i>dstcluster</i>	The name of the destination cluster. For a containerized deployment, set this to the target data plane (For example, <i>dstcluster=data</i> )
<i>srcdali</i>	Optional. The IP address of the source Dali server, if different from the destination Dali (associated with the ESP Server specified in the <i>server</i> option).
<i>srcusername</i>	Optional. The username to use to access the <i>srcdali</i> . If omitted, the General Options <i>username</i> is used.
<i>srcpassword</i>	Optional. The password to use to access the <i>srcdali</i> . If omitted, the General Options <i>password</i> is used.
<i>preservecompression</i>	Optional. A boolean flag (0   1) indicating whether to preserve the compression of the source file. If omitted, the default is 1.
<i>ensure</i>	Optional. Copies logical file, but does not copy file parts if they already exist. Default is FALSE.

Example:

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

## Remove Operations:

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

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

<i>name</i>	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.

This also allows you to restore a superfile whose information was previously exported using the savexml action. This is especially useful in a Cloud implementation where files are stored in a bucket until a new instance is started.

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

srcxml	The path and name of the source XML file containing exported logical or superfile information (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 <i>srcdali</i> . If omitted, the General Options <i>username</i> is used.
srcpassword	Optional. The password to use to access the <i>srcdali</i> . If omitted, the General Options <i>password</i> is used.

Example:

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



## 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:

<i>superfile</i>	The logical name of the superfile.
<i>subfiles</i>	A comma-delimited list of the logical names of files to add to the superfile. There must be no spaces between the names.
<i>before</i>	Optional. The logical name of the subfile to follow the added <i>subfiles</i> . If omitted, the <i>subfiles</i> 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:

<i>superfile</i>	The logical name of the superfile.
<i>subfiles</i>	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.
<i>delete</i>	Optional. A boolean flag (1   0) indicating whether to physically delete the <i>subfiles</i> 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
```

## Copysuper Operations:

The **copysuper** operation copies a superfile from one cluster to another.

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

<i>srcname=&lt;source-super-name&gt;</i>	The logical name of the source superfile.
<i>dstname=&lt;destination-super-name&gt;</i>	The logical name of the destination superfile.
<i>dstcluster=&lt;cluster-name&gt;</i>	The name of the destination cluster. For a containerized deployment, set this to the target data plane (For example, <i>dstcluster=data</i> )
<i>srcdali</i>	The hostname or IP of the source Dali server.
<i>srcusername</i>	Optional, The username to use for the source environment.

<i>srcpassword</i>	Optional, The password to use for the source environment.
--------------------	---

Example:

```
dfuplus action=copysuper srcname=jd::super1 dstname=jd::super2 dstcluster=mythor srcdali=.
```

## 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:

<i>superfile</i>	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:

<i>wuid</i>	The workunit identifier of the workunit.
<i>limit</i>	For publisher workunits, this limits the number of child workunits to list. If set to 0, then only the parent status is returned. Default is 1000.

Examples:

```
dfuplus action=status wuid=D20220816-135243
dfuplus action=status limit=300 wuid=P20230301-123456
dfuplus action=status limit=0 wuid=P20230301-987654
```

## Abort Operations:

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

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

<i>wuid</i>	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:

<i>wuid</i>	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.

This feature also allows you to export the metadata from a superfile and then use it later to restore a superfile. This is especially useful in an Cloud implementation where files are stored in a bucket until a new instance is started.

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

*srcname* The logical name of the source file.

<i>srcname</i>	The logical name of the source file. This can be the logical name of a superfile.
<i>dstxml</i>	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="/var/lib/HPCCSystems/hpcc-data/thor/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:

<i>event</i>	The name of the user-defined event to trigger. This is used as the first parameter of the ECL EVENT function.
<i>lfn</i>	Optional. The name of the logical file in the DFU to look for. Using this option precludes using the <i>ip</i> , <i>file</i> , and <i>sub</i> options.
<i>ip</i>	Optional. The IP address or name of the server on which the physical file will reside. This may be omitted if the <i>file</i> option contains a full URL.
<i>file</i>	Optional. The fully qualified path of the physical file to look for. This may contain wildcard characters (* and ?).
<i>sub</i>	Optional. Specifies searching subdirectories for the physical file if the <i>file</i> option contains wildcard characters (* and ?).
<i>shotlimit</i>	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 possibility, 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=/var/lib/HPCCSystems/mydropzone/arr.txt
dfuplus action=monitor event=MyEvent ip=10.150.10.75
        file=/var/lib/HPCCSystems/mydropzone/* shotlimit=-1 sub=1
dfuplus action=monitor event=MyEvent file=//10.15.13.21/var/lib/HPCCSystems/mydropzone/*.txt
dfuplus action=monitor event=MyEvent lfn=RTTEMP::OUT::MyFile
```

## Listhistory Operations:

The listhistory operation returns the history metadata in a logical file.

History metadata is created from a copy, remote copy, or spray operation.

<i>lfn</i>	The logical file name of the source file
<i>outformat</i>	Optional. The format of the result. Valid options are: csv, xml, json, or ascii. The default is xml.
<i>csvheader</i>	Optional. A boolean flag (0   1) indicating whether to include header information in the first row when outputting in csv format.

**Example:**

```
dfuplus action=listhistory lfn=progguide::exampledata::accounts
// this results in the following XML file:
<History>
  <Origin ip="127.0.0.1"
    name="accounts"
    operation="DFUcopy"
    owner="EmilyKate"
    path="/var/lib/HPCCSystems/hpcc-data/thor/progguide/exampledata/"
    timestamp="2017-05-11T16:47:32"
    workunit="W20170503-143100"/>
</History>
```

## Erasehistory Operations:

The erasehistory operation removes the history metadata from a logical file.

History metadata is created from a copy, remote copy, or spray operation.

**Note:** If LDAP authentication is enabled on the system, you must have FULL permission for DFUAccess in order to erase history. See the HPCC Systems Administrator's Guide for details.

<i>lfn</i>	The logical file name of the source file
<i>backup</i>	Optional. A boolean flag (0   1) indicating whether to write the history to a file before erasing it. Default is 1 (enable).
<i>dstxml</i>	The logical name of the destination XML file. Required if backup is set to 1,

**Example:**

```
dfuplus action=erasehistory lfn=progguide::exampledata::accounts_copy dstxml=c:\temp\jim.xml

// this removes the history metadata from the file and writes an XML file containing the following:
<History>
  <Origin ip="127.0.0.1"
    name="accounts"
    operation="DFUcopy"
    owner="EmilyKate"
    path="/var/lib/HPCCSystems/hpcc-data/thor/progguide/exampledata/"
    timestamp="2017-05-11T16:47:32"
    workunit="W20170503-143100"/>
</History>
```

# ESDL Command Line Interface

## The ESDL Command Syntax

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

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

## esdl xml

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

<i>filename.ecm</i>	The file containing the ESDL definitions
<i>-r --recursive</i>	process all includes
<i>-I, --include-path</i>	Locations to look for included ESDL files. They can be absolute or relative paths. If you need to specify multiple directories, you can use multiple <i>-I</i> options or use a single entry with the directories separated with the environment separator character. For Linux, use a colon (:) and for Windows, use a semi-colon (;). The paths can also be set using an environment variable--ESDL_INCLUDE_PATH.
<i>-v --verbose</i>	display verbose information
<i>-?/-h/--help</i>	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 sourcePath outputPath [options].**

<i>sourcePath</i>	The absolute path to the ESDL Definition file containing the EsdlService definition for the service.
<i>outputPath</i>	The absolute path to the location where ECL output is to be written.
<i>-x, --expandedxml</i>	Output expanded XML files.
<i>--includes</i>	If present, process all included files.
<i>--rollup</i>	If present, rollup all processed includes to a single ECL output file.
<i>-cde</i>	Specifies the HPCC Systems Component files directory (location of xslt files).
<i>--ecl-imports</i>	Comma-delimited import list to be attached to the output ECL. Each entry generates a corresponding IMPORT statement.
<i>--ecl-header</i>	Text to include in header or target (generated) file (must be valid ECL).
<i>-I, --include-path</i>	Locations to look for included ESDL files. They can be absolute or relative paths. If you need to specify multiple directories, you can use multiple -I options or use a single entry with the directories separated with the environment separator character. For Linux, use a colon (:) and for Windows, use a semi-colon (;). The paths can also be set using an environment variable--ESDL_INCLUDE_PATH.
Output	(sourcePath outputPath>)/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 sourcePath serviceName [options]**

<i>sourcePath</i>	The absolute path to the ESDL Definition file containing the EsdlService definition for the service.
<i>serviceName</i>	Name of ESDL Service defined in the given ESDL file.
<i>--version &lt;version number&gt;</i>	Constrain to interface version
<i>--method &lt;method name&gt;[;&lt;method name&gt;]*</i>	Constrain to list of specific method(s)
<i>--xslt &lt;xslt file path&gt;</i>	Path to '/xslt/esxdl2xsd.xslt' file to transform EsdlDef to XSD
<i>--preprocess-output &lt;raw output directory&gt; :</i>	Output preprocessed XML file to specified directory before applying XSLT transform
<i>--annotate &lt;all   none&gt;</i>	Flag turning on either all annotations or none. By default, annotations are generated for Enumerations. Setting the flag to 'none' will disable those as well. Setting it to 'all' enables additional annotations such as collapsed, cols, form_ui, html_head and rows.
<i>--noopt</i>	Turns off the enforcement of 'optional' attributes on elements. If no -noopt is specified then all elements with an 'optional' are included in the output. By default 'optional' filtering is enforced.
<i>-opt,--optional &lt;param value&gt;</i>	Value to use for optional tag filter when gathering dependencies. For example, passing 'internal' when some ESDL definition objects have the attribute optional("internal") ensures they appear in the XSD, otherwise they'd be filtered out
<i>-tns,--target-namespace &lt;target namespace&gt;</i>	The target namespace passed to the transform via the parameter 'tnsParam' used for the final output of the XSD.
<i>-n &lt;int&gt; .</i>	Number of times to run transform after loading XSLT. Defaults to 1
<i>--show-inheritance</i>	Turns off the collapse feature. Collapsing optimizes the XML output to strip out structures only used for inheritance, and collapses their elements into their child. That simplifies the stylesheet. By default this option is on
<i>--no-arrayof</i>	Suppresses the use of the arrayOf element. arrayOf optimizes the XML output to include 'ArrayOf...' structure definitions for those EsdlArray elements with no item_tag attribute. Works in conjunction with an optimized stylesheet that doesn't generate these itself. This defaults to on.
<i>-v --verbose</i>	display verbose information
<i>-?/-h/--help</i>	show usage page
Output	(srcdir <outdir>)/filename.ecf

This generates XSD from the ESDL definition.

Examples:

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

## esdl wsdli

**esdl wsdli sourcePath serviceName [options]**

<i>sourcePath</i>	The absolute path to the ESDL Definition file containing the EsdlService definition for the service.
<i>serviceName</i>	Name of ESDL Service defined in the given ESDL file.
<i>--version &lt;version number&gt;</i>	Constrain to interface version
<i>--method &lt;method name&gt;[;&lt;method name&gt;]*</i>	Constrain to list of specific method(s)
<i>--xslt &lt;xslt file path&gt;</i>	Path to '/xslt/esxdli2xsd.xslt' file to transform EsdlDef to XSD
<i>--preprocess-output &lt;raw output directory&gt; :</i>	Output preprocessed XML file to specified directory before applying XSLT transform
<i>--annotate &lt;all   none&gt;</i>	Flag turning on either all annotations or none. By default, annotations are generated for Enumerations. Setting the flag to 'none' will disable those as well. Setting it to 'all' enables additional annotations such as collapsed, cols, form_ui, html_head and rows.
<i>--noopt</i>	Turns off the enforcement of 'optional' attributes on elements. If no -noopt is specified then all elements with an 'optional' are included in the output. By default 'optional' filtering is enforced.
<i>-opt,--optional &lt;param value&gt;</i>	Value to use for optional tag filter when gathering dependencies. For example, passing 'internal' when some ESDL definition objects have the attribute optional("internal") ensures they appear in the XSD, otherwise they'd be filtered out
<i>-tns,--target-namespace &lt;target namespace&gt;</i>	The target namespace passed to the transform via the parameter 'tnsParam' used for the final output of the XSD.
<i>-n &lt;int&gt; .</i>	Number of times to run transform after loading XSLT. Defaults to 1
<i>--show-inheritance</i>	Turns off the collapse feature. Collapsing optimizes the XML output to strip out structures only used for inheritance, and collapses their elements into their child. That simplifies the stylesheet. By default this option is on
<i>--no-arrayof</i>	Suppresses the use of the arrayOf element. arrayOf optimizes the XML output to include 'ArrayOf...' structure definitions for those EsdlArray elements with no item_tag attribute. Works in conjunction with an optimized stylesheet that doesn't generate these itself. This defaults to on.
<i>--wsdladdress</i>	Defines the output WSDL file's location address
<i>-v/--verbose</i>	display verbose information
<i>-?/-h/--help</i>	show usage page
Output	(srcdir <outdir>)/filename.eci

This generates WSDL from ESDL definition.

Examples:

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

## esdl publish

**esdl publish <filename.(ecm|esdl|xml)> <servicename> [options]**

filename	The ESDL (*.ecm, *.esdl, or *.xml) file containing the service definitions.
servicename	The name of the service to publish. Optional if the ESDL definition contains only one service.
--overwrite	Overwrite the latest version of this ESDL Definition
-I, --include-path	Locations to look for included ESDL files. They can be absolute or relative paths. If you need to specify multiple directories, you can use multiple -I options or use a single entry with the directories separated with the environment separator character. For Linux, use a colon (:) and for Windows, use a semi-colon (;). The paths can also be set using an environment variable--ESDL_INCLUDE_PATH.
-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)
--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.ecm MathService -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 <ESDLDefinitionID> [options]**

ESDLDefinitionID	The ID 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** <TargetESPProcessName> <TargetESPBindingPort> <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) Required if ESDL definition contains multiple services
--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 either the port on which this service is configured to run (ESP Binding) or 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="<RoxieIPRange>:9876/path?param=value" user="me" password="mypw" />
  <Method name="myMthd2" url="<RoxieIPRange>:9876/path?param=value" user="me" password="mypw" />
</Methods>
```

### Example:

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

## Configuring ESDL binding methods

The ESDL binding methods can optionally provide context information to the target ECL query. The way this information is configured, is by appending child elements to the Method (<Method>...</Method>) portion of the ESDL Binding.

For example, the following XML provides a sample ESDL Binding.

```
<Methods>
  <Method name="AddThis" url="<RoxieIPRange>:9876" querytype="roxie" queryname="AddThis"/>
</Methods>
```

If this Method requires context information, for example about gateways, then you could include the Gateways Structure (<Gateways>...</Gateways>) depicted as follows.

```
<Methods>
  <Method name="AddThis" url="<RoxieIPRange>:9876" querytype="roxie" queryname="AddThis">
    <!--Optional Method Context Information start-->
    <Gateways>
      <Gateway name="mygateway" url="1.1.1.1:2222/someservice/somemethod/">
      <Gateway name="anothergateway" url="2.2.2.2:9999/someservice/somemethod/">
    </Gateways>
    <!--Optional Method Context Information end-->
  </Method>
</Methods>
```

The DESDL ESP does not pose any restrictions on the layout of this information, only that it is valid XML. This provides the flexibility to include context information in any valid XML format.

Roxie (query) ECL developers need to decide what information they will need from the ESP request and design how that information is laid-out in the ESP request and ESDL binding configuration.

In the following example, every "AddThis" request processed by the ESP and sent to Roxie would contain the sample gateway information in the request context.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
  <roxie.AddThis>
    <Context>
      <Row>
        <Common>
          <ESP>
            <ServiceName>wsmath</ServiceName>
            <Config>
              <Method name="AddThis" url="<RoxieIPRange>:9876" querytype="roxie" queryname="AddThis">
                <Gateways>
                  <Gateway name="mygateway" url="1.1.1.1:2222/someservice/somemethod/">
                  <Gateway name="anothergateway" url="2.2.2.2:9999/someservice/somemethod/">
                </Gateways>
              </Method>
            </Config>
          </ESP>
          <TransactionId>sometrxd</TransactionId>
        </Common>
      </Row>
    </Context>
    <AddThisRequest>
      <Row>
        <Number1>34</Number1>
        <Number2>232</Number2>
      </Row>
    </AddThisRequest>
  </roxie.AddThis>
</soap:Body>
</soap:Envelope>
```

The ECL query consumes this information and is free to do whatever it needs to with it. In some instances, the query needs to send a request to a gateway in order to properly process the current request. It can interrogate the context information for the appropriate gateway's connection information, then use that information to create the actual gateway request connection.



## Configuring ESDL binding for Proxy Mode methods

You can specify that ESDL service methods be proxied to another ESP instance. Set up the proxy in the dynamic configuration associated with the dESDL service.

Under the Methods tag where you would add Method tags, you can also add Proxy tags, as shown here:

```
<Methods>
  <Method name="myMethod" url="http://10.45.22.1:292/somepath" />
  <Method name="myMethod2" url="http://10.45.22.1:292/somepath" />
  <Proxy method="myMethod3" forwardTo="http://10.45.22.1:292" />
  <Proxy method="myWild*" forwardTo="http://10.45.22.1:292" />
</Methods>
```

The Proxy tag also supports wildcards:

```
<Proxy method="myWild*" forwardTo="http://10.45.22.1:292" />
```

This example binds all methods matching the pattern: myWild\*

## 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 <ESPBindingID> [options]**

ESPBindingID	The ESDL Binding ID
-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 unbind ESDL service based bindings.

To unbind a given ESDL binding, provide the ESP process name and the ESDL binding ID

Available ESDL bindings to unbind can be found using the "esdl list-bindings" command

### Example:

```
esdl unbind-service myesp.8003.MathService
```

## esdl bind-method

**esdl bind-method <TargetESDLBindingID> <TargetMethodName> [options]**

TargetESDLBindingID	The id of the target ESDL binding (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://<RoxieIPRange>:9876/path?param=value" user="me" password="mypw"/>
  <Method name="myMthd2" url="http://<RoxieIPRange>:9876/path?param=value" user="me" password="mypw"/>
</Methods>
```

### Example:

```
esdl bind-method myesp.8003.MathService AddThis --config myMethods.xml
```

## esdl unbind-method

**esdl unbind-method <ESDLBindingID> <MethodName> [options]**

ESDLBindingID	The ID of the ESDL Binding
MethodName	The name of the target method (must exist in the service 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 unbind a method configuration associated with a given ESDL binding. To unbind a method, provide the ID of the ESDL binding and the name of the method you are unbinding.

### Example:

```
esdl unbind-method myesp.8003.MathService AddThis
```

## esdl get-binding

**esdl get-binding <ESDLBindingId> [options]**

ESDLBindingId	The target ESDL binding id <ESPPProcessName>.<Port>.<ServiceName>
-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 ID of the ESDL binding, which is normally in the format <ESPPProcessName>.<Port>.<ServiceName>

### Example:

```
esdl get-binding myesp.8003.MathService -s nnn.nnn.nnn.nnn -p 8010
```